



# Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

## ► To cite this version:

Bruno Blanchet. Composition Theorems for CryptoVerif and Application to TLS 1.3. [Research Report] RR-9171, Inria Paris. 2018, pp.67. hal-01764527

**HAL Id: hal-01764527**

**<https://inria.hal.science/hal-01764527>**

Submitted on 12 Apr 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

**RESEARCH  
REPORT**

**N° 9171**

April 2018

Project-Team Prosecco





# Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

Project-Team Prosecco

Research Report n° 9171 — April 2018 — 67 pages

**Abstract:** We present composition theorems for security protocols, to compose a key exchange protocol and a symmetric-key protocol that uses the exchanged key. Our results rely on the computational model of cryptography and are stated in the framework of the tool CryptoVerif. They support key exchange protocols that guarantee injective or non-injective authentication. They also allow random oracles shared between the composed protocols. To our knowledge, they are the first composition theorems for key exchange stated for a computational protocol verification tool, and also the first to allow such flexibility.

As a case study, we apply our composition theorems to a proof of TLS 1.3 Draft-18. This work fills a gap in a previous paper that informally claims a compositional proof of TLS 1.3, without formally justifying it.

**Key-words:** composition, security protocols, verification, computational model, TLS

**RESEARCH CENTRE  
PARIS**

2 rue Simone Iff  
CS 42112  
75589 Paris Cedex 12

## Théorèmes de composition pour CryptoVerif et application à TLS 1.3

**Résumé :** Nous présentons des théorèmes de composition pour les protocoles cryptographiques, pour composer un protocole d'échange de clés et un protocole à clé symétrique qui utilise la clé échangée. Nos résultats reposent sur le modèle calculatoire de la cryptographie et sont formulés dans le cadre de l'outil CryptoVerif. Ils autorisent des protocoles d'échange de clés qui garantissent l'authentification injective ou non-injective. Ils autorisent aussi le partage d'oracles aléatoires entre les protocoles composés. À notre connaissance, ils sont les premiers théorèmes de composition pour l'échange de clés formulés pour un outil de vérification de protocole dans le modèle calculatoire, et aussi les premiers à autoriser une telle flexibilité.

Comme étude de cas, nous appliquons nos théorèmes de composition à une preuve de TLS 1.3 brouillon 18. Ce travail fournit un élément manquant dans un article précédent qui donne informellement une preuve compositionnelle de TLS 1.3, sans la justifier formellement.

**Mots-clés :** composition, protocoles cryptographiques, vérification, modèle calculatoire, TLS

# 1 Introduction

The proof of security protocols is notoriously difficult. In particular, when security protocols grow in size, the complexity of their proof increases, and quickly becomes unmanageable. Composition theorems are essential to tackle this problem: they allow one to prove small pieces of the considered protocol, and to compose these results in order to obtain a proof of the full protocol.

In this paper, we consider the standard game-based computational model of cryptography, and we focus on the composition between a key exchange protocol and a symmetric key protocol that uses the key provided by the key exchange. We assume that the key exchange protocol runs between two participants  $A$  and  $B$  and is secure: the provided key is secret in the sense that keys provided in several sessions are indistinguishable from independent random keys; the protocol provides authentication of  $A$  to  $B$ , defined based on session identifiers; and  $A$  executes at most one session with a given session identifier. Then, we prove that the security properties of the symmetric key protocol carry over to the composed protocol. Moreover, we also prove that the security properties of the key exchange protocol are preserved in the composed protocol (except for the secrecy of the key on which we perform the composition). This point is important to be able to perform several compositions one after the other. We have two variants of our composition theorem: one in which authentication is injective, that is, each execution of  $B$  corresponds to a distinct execution of  $A$ ; and one in which authentication is non-injective, that is, several executions of  $B$  may correspond to the same execution of  $A$ .

An originality of our composition theorems is that they are stated within the framework of a protocol verification tool, CryptoVerif [11, 12, 14], available at <http://cryptoverif.inria.fr>. We use the language of CryptoVerif to represent the cryptographic games; we use security properties proved by CryptoVerif as assumptions of our composition theorems. Therefore, we can easily use CryptoVerif to mechanize the proof of the protocol pieces that we compose.

Using such a framework to state composition theorems has several advantages. It strengthens the abilities of CryptoVerif: thanks to the composition theorems, we can obtain security proofs for bigger protocols. Moreover, CryptoVerif does not support loops. If we can break a protocol with loops into pieces without loops, we can verify them using CryptoVerif, and prove security of the whole protocol with loops by iteratively composing these pieces. Finally, CryptoVerif provides a rigorous framework for stating the composition theorems: it provides a formal language for games and for security properties, with a formal semantics. The hypotheses and conclusions of the composition theorems can then be stated precisely and concisely in that framework.

As a case study, we apply our composition theorems to TLS 1.3. More precisely, we revisit a previous analysis of TLS 1.3 Draft 18 by Bhargavan et al. [9, 10]. This analysis splits TLS 1.3 into 3 pieces: the initial handshake, the handshake with pre-shared key, and the record protocol, and claims that security of TLS 1.3 can be obtained by composing these 3 pieces. However, it does not justify this composition formally. Our work fills this gap. TLS 1.3 is particularly well-suited as a case study to illustrate the power of our composition results. First, it is an important protocol, which is currently being standardized. The current draft, Draft 28 [35], is now final, and not very different from Draft 18. We expect that our composition results would apply in the same way to Draft 28, though the CryptoVerif models of the protocol pieces would require minor changes. Second, TLS 1.3 is well-designed to allow composition: the handshake produces traffic secrets used by the record protocol as well as a resumption secret used as a pre-shared key by the next handshake. The protocol pieces are cleanly separated, so that the only common secret between them is the symmetric key on which we perform the composition. Third, TLS 1.3 includes loops: it allows an unbounded number of handshakes with pre-shared key and an unbounded number of key updates in the record protocol. Therefore, it cannot be analyzed as a whole by CryptoVerif. The composition results allow us to break these loops and obtain security results for the full

protocol. Finally, TLS 1.3 includes a variety of compositions, and we provide theorems for all of them. While most compositions use a key exchange that provides injective authentication, TLS also includes 0-RTT (Round Trip Time) data, that is, data that the client sends to the server immediately after the first message of TLS (`ClientHello`). Such data can be replayed, and the corresponding key exchange only provides non-injective authentication. Furthermore, we also have to deal with altered `ClientHello` messages; in this case, only the server has the corresponding key, and that requires a variant of the composition result. Finally, for key updates in the record protocol, the key is simply computed from the previous key without a proper key exchange, so we can use a much simpler composition theorem in this case.

**Related Work** The line of work closest to ours is that of [16, 27, 29]. Brzuska et al. [16] prove a composition result similar to ours, in an informal game-based framework. Fischlin et al. [27, 29] extend this framework to multi-stage key exchange protocols, in which parties can establish multiple keys in different stages and use these keys between stages. They apply their results to the proof of QUIC [29] and of TLS 1.3 Draft 05, Draft DH [27], and Draft 10 [28]. In addition to recasting the composition result in *CryptoVerif*, we extend it in several ways. We prove that security properties of the key exchange protocol are preserved in the composition, so we can compose again using other keys. This point appears in [29, Remark, page 16 of the full version] without proof. We allow the composed protocols to share random oracles; this point does not appear in [16, 27, 29]. We prove a composition theorem for protocols that provide non-injective authentication, used for 0-RTT data in TLS 1.3. Although we do not consider several stages explicitly, our composition theorems support most compositions allowed by the multi-stage framework. (We detail the comparison in Section 5.4.)

Brzuska et al. [15] prove a composition theorem that allows the key exchange protocol to already use the key provided to the symmetric-key protocol, for example for key confirmation. TLS 1.3 is designed so that the same key is never both used in the key exchange and provided to the next protocol, so we did not need such a composition result in our case study. We believe that such a result could also be proved in our framework if desired.

Canetti and Krawczyk [19] prove security of the composition of a key exchange protocol with specific symmetric-key protocols that use MACs to achieve an authenticated channel or encrypt-then-MAC to achieve a secure channel.

Barthe et al. [4] develop generic proofs of reduction for one-round key-exchange protocols, such as Naxos and HMQV, in *EasyCrypt*. *EasyCrypt* is an interactive theorem prover specialized for building game-based security proofs. Thus, their approach provides another way of introducing modularity in machine-checked proofs of security.

Universal composability (UC) [17, 18, 20, 32] is a framework that allows to compose protocols. However, proving UC-security requires stronger properties than the game-based framework that we use ([16, Appendix A] details limitations of the UC framework). Delaune et al. [25] present a simulation-based framework that is an analogue of UC in the symbolic (Dolev-Yao) model of cryptography.

Composition theorems have also been proved in the Dolev-Yao model. Many of these theorems deal with the parallel composition of protocols that share secrets, for trace properties [22, 31], for resistance against guessing attacks for protocols that share passwords [26], and for privacy properties [2], using disjointness assumptions such as tagging or disjoint primitives to guarantee the independence of the protocols. Other results [3, 21] allow sequential composition, in particular the composition of a key exchange protocol with a symmetric-key protocol that uses the exchanged key. Ciobăcă et al. [21] consider trace properties, while Arapinis et al. [3] extend the result to processes with else branches, to private channels, and to privacy properties. Mödersheim et al. [30, 33, 34] define notions of security for channels (insecure, authentic, confidential, secure),

and prove composition results between protocols that establish such channels and protocols that use them. They also rely on the Dolev-Yao model and use disjointness assumptions. We believe that the computational model has two advantages: it is more realistic than the Dolev-Yao model and the computational definitions compose nicely, so that we can avoid disjointness assumptions.

Protocol composition logic [23,24] is a logic for proving security protocols that allows sequential and parallel composition. It was initially designed in the Dolev-Yao model [23] and adapted to the computational model [24].

**Outline** The next section provides a minimal reminder of the CryptoVerif framework. Section 3 presents the structure of our proof of TLS, so that we can use it as a motivation for the composition theorems. Section 4 presents a very simple composition theorem, used for key updates in TLS 1.3, as a warm-up. Section 5 presents our main composition theorems. Section 6 summarizes their application to TLS, and Section 7 concludes. The appendix provides the proofs of all results and details on the TLS case study.

## 2 A Short Reminder on CryptoVerif

**Processes, contexts, adversaries** CryptoVerif mechanizes proofs by sequences of games, similar to those written on paper by cryptographers [8,36]. It represents protocols and cryptographic games in a probabilistic process calculus. We refer the reader to [12,14] for details on this process calculus. We explain the necessary constructs as they appear.

We use  $P, Q$  for *processes*. A *context*  $C$  is a process with one or several holes  $[]$ . We write  $C[P_1, \dots, P_n]$  for the process obtained by replacing the holes of  $C$  with  $P_1, \dots, P_n$  respectively. An *evaluation context* is a context with one hole, generated by the following grammar:

$C ::=$	evaluation context
$[]$	hole
$\mathbf{newChannel} \ c; C$	channel restriction
$Q \mid C$	parallel composition
$C \mid Q$	parallel composition

The channel restriction  $\mathbf{newChannel} \ c; Q$  restricts the channel name  $c$ , so that communications on this channel can occur only inside  $Q$ , and cannot be received outside  $Q$  or sent from outside  $Q$ . The parallel composition  $Q_1 \mid Q_2$  makes simultaneously available the processes defined in  $Q_1$  and  $Q_2$ . We use evaluation contexts to represent *adversaries*.

**Indistinguishability** A process can execute events, by two constructs: **event**  $e(M_1, \dots, M_n)$  executes event  $e$  with arguments  $M_1, \dots, M_n$ , and **event\_abort**  $e$  executes event  $e$  without argument and aborts the game. After finishing execution of a process, the system produces two results: the sequence of executed events  $\mathcal{E}$ , and the information whether the game aborted ( $a = \mathbf{abort}$ , that is, executed **event\_abort**) or terminated normally ( $a = 0$ ). These events and result can be used to distinguish games, so we introduce an additional algorithm, a *distinguisher*  $D$  that takes as input the sequence of events  $\mathcal{E}$  and the result  $a$ , and returns **true** or **false**. We write  $\Pr[Q : D]$  for the probability that the process  $Q$  executes events  $\mathcal{E}$  and returns a result  $a$  such that  $D(\mathcal{E}, a) = \mathbf{true}$ .

**Definition 1** (Indistinguishability). *We write  $Q \approx_p^V Q'$  when, for all evaluation contexts  $C$  acceptable for  $Q$  and  $Q'$  with public variables  $V$  and all distinguishers  $D$  that run in time at most  $t_D$ ,  $|\Pr[C[Q] : D] - \Pr[C[Q'] : D]| \leq p(C, t_D)$ .*



Intuitively,  $Q \approx_p^V Q'$  means that an adversary has probability at most  $p$  of distinguishing  $Q$  from  $Q'$ , when it can read the variables in the set  $V$ . When  $V$  is empty, we omit it. The probability  $p$  may depend on many parameters coming from the context  $C$ , that is why  $p$  takes as arguments the whole context  $C$  and the runtime of  $D$ . CryptoVerif always expresses the probabilities as formulas in which the only parameters that come from the context  $C$  are the maximum runtime of  $C$ , the maximum number of times  $C$  may send a message to each subprocess in  $Q$  (resp.  $Q'$ ), and the lengths of bitstrings. This property allows us to simplify probability formulas by abstracting away the precise context they use and retaining only the useful parameters. We denote by  $t_C$  the maximum runtime of  $C$ , and use the same notation for processes  $P$ ,  $Q$ , terms  $M$ , and functions  $f$ . As usual in cryptographic proofs, we ignore very small runtimes.

The condition that  $C$  is acceptable for  $Q$  and  $Q'$  with public variables  $V$  is a technical condition that ensures that  $C[Q]$  and  $C[Q']$  are well-formed. The public variables  $V$  are the variables of  $Q$  and  $Q'$  that  $C$  is allowed to read.

Indistinguishability is reflexive ( $Q \approx_0^V Q$ ), symmetric (if  $Q \approx_p^V Q'$ , then  $Q' \approx_p^V Q$ ) and transitive (if  $Q \approx_p^V Q'$  and  $Q' \approx_{p'}^V Q''$ , then  $Q \approx_{p+p'}^V Q''$ ). Moreover,  $Q \approx_p^V Q'$  implies  $C[Q] \approx_{p'}^{V'} C[Q']$  for all evaluation contexts  $C$  acceptable for  $Q$  and  $Q'$  with public variables  $V$  and all  $V' \subseteq V \cup \text{var}(C)$ , where  $\text{var}(C)$  is the set of variables of  $C$  and  $p'(C', t_D) = p(C'[C[]], t_D)$ .

**Secrecy** Intuitively, in CryptoVerif, secrecy means that the adversary cannot distinguish between the secrets and independent random values. This definition corresponds to the “real-or-random” definition of security [1]. As shown in [1], this notion is stronger than the one in which the adversary performs a single test query and some reveal queries. We recall the definition of secrecy in CryptoVerif given in [13,14]. Let us first explain CryptoVerif constructs used in this definition. The *replication*  $!^{i \leq n} Q$  represents  $n$  copies of the process  $Q$  in parallel, indexed by  $i \in [1, n]$ , where  $n$  is named a *replication bound*. The *current replication indices* at a certain program point are the indices  $i$  of replications above that program point. In CryptoVerif, all variables defined under a replication are implicitly arrays indexed by the current replication indices: if  $Q$  defines a variable  $x$  under  $!^{i \leq n}$ , the value of  $x$  is in fact stored in  $x[i]$ . The definition of  $x$  is executed at most once for each  $i$ , so that all values of  $x$  are stored in distinct array cells. When a variable is accessed with current replication indices, we omit the indices, writing  $x$  for  $x[i]$ . The **find** construct reads these array cells: **find**  $u = i' \leq n$  **suchthat**  $\text{defined}(x_1[i'], \dots, x_m[i']) \wedge M$  **then**  $P$  **else**  $P'$  looks for an index  $i' \in [1, n]$  such that  $x_1[i'], \dots, x_m[i']$  are defined and  $M$  is true. When such an index is found, it is stored in  $u$ , and process  $P$  is executed. Otherwise, process  $P'$  is executed. The term  $M$  may refer to  $x_1[i'], \dots, x_m[i']$  and the process  $P$  may refer to  $x_1[u], \dots, x_m[u]$  since these variables are guaranteed to be defined. The *input*  $c[i](x : T); P$  receives a message on channel  $c[i]$ . If this message is in the set of bitstrings  $T$  ( $T$  stands for “type”), it is stored in  $x$ , and  $P$  is executed. Otherwise, the process blocks. The channel  $c[i]$  consists of a channel name  $c$  and indices, here  $i$ . Very often, these indices are the current replication indices at the input: the sender can then tell precisely to which copy of the process the message should be sent. Similarly, the *output*  $\overline{c[i]}(M); Q$  sends message  $M$  on channel  $c[i]$ . After the output, the control is passed to the receiver process, which continues execution. The process  $Q$  that follows the output consists of inputs, possibly under replications and parallel compositions; these inputs will be executed when a message is sent to them. Finally, the *restriction* **new**  $y : T; P$  chooses uniformly a random element of  $T$ , stores it in  $y$ , and executes  $P$ .

We use  $\tilde{u}$  as an abbreviation for a sequence of variables:  $\tilde{u} = u_1, \dots, u_m$ . We write  $\tilde{u} \leq \tilde{n}$  for  $u_1 : [1, n_1], \dots, u_m : [1, n_m]$  when  $\tilde{u} = u_1, \dots, u_m$  and  $\tilde{n} = n_1, \dots, n_m$ . We say that a variable is defined under replications  $!^{i_1 \leq \tilde{n}} \dots !^{i_m \leq \tilde{n}}$  when  $\tilde{i} = i_1, \dots, i_m$ ,  $\tilde{n} = n_1, \dots, n_m$ , and it is defined under replications  $!^{i_1 \leq n_1} \dots !^{i_m \leq n_m}$ . (There may be other instructions between these replications.) We

define that a context has replications  $!^{\tilde{i} \leq \tilde{n}}$  above the hole in a similar way. When  $\tilde{n} = n_1, \dots, n_m$ , we define  $\prod \tilde{n} = n_1 \times \dots \times n_m$ .

**Definition 2** (Secrecy). *Let  $x$  and  $V$  be such that  $x \notin V$ . Suppose that the variable  $x$  has type  $T$  and is defined under replications  $!^{\tilde{i} \leq \tilde{n}}$  in  $Q$ . Let*

$$\begin{aligned} R_x = & c_{s0}(); \text{new } b : \text{bool}; \overline{c_{s0}}(); \\ & (!^{i_s \leq n_s} c_s[i_s](\tilde{u} \leq \tilde{n}); \text{if defined}(x[\tilde{u}]) \text{ then} \\ & \quad \text{if } b \text{ then } \overline{c_s[i_s]}(x[\tilde{u}]) \text{ else} \\ & \quad \text{find } u'_s = i'_s \leq n_s \text{ suchthat defined}(y[i'_s], \tilde{u}[i'_s]) \wedge \tilde{u}[i'_s] = \tilde{u} \text{ then } \overline{c_s[i_s]}(y[u'_s]) \text{ else} \\ & \quad \text{new } y : T; \overline{c_s[i_s]}(y) \\ & \quad | c'_s(b' : \text{bool}); \text{if } b = b' \text{ then event\_abort } S \text{ else event\_abort } \bar{S}) \end{aligned}$$

where the channels  $c_{s0}, c_s, c'_s$ , the variables  $\tilde{u}, u'_s, y, b, b'$ , and the events  $S, \bar{S}$  do not occur in  $Q$ .

The process  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$  when, for all evaluation contexts  $C$  acceptable for  $Q \mid R_x$  with public variables  $V$  that do not contain  $S$  nor  $\bar{S}$ ,  $\Pr[C[Q \mid R_x] : S] - \Pr[C[Q \mid R_x] : \bar{S}] \leq p(C)$ .

The process  $R_x$  chooses a random bit  $b$ , and then allows the adversary to query the variable  $x$ : if the adversary sends indices  $\tilde{u}$  on channel  $c_s[i_s]$ , and  $x[\tilde{u}]$  is defined, then the process  $R_x$  replies with the value of  $x[\tilde{u}]$  when  $b$  is true, and with a random value when  $b$  is false. The **find** in  $R_x$  makes sure that, if the indices  $\tilde{u}$  have already been queried, then the previous reply is sent; otherwise, a fresh random value  $y$  is chosen in the type  $T$  of  $x$  by **new**  $y : T$ , and sent as a reply. The replication  $!^{i_s \leq n_s}$  in  $R_x$  allows the adversary to perform at most  $n_s$  such queries;  $n_s$  is chosen large enough so that it is not a limitation. Finally, the adversary sends on channel  $c'_s$  its guess  $b'$  for the bit  $b$ . If the guess is correct ( $b' = b$ ), then the process  $R_x$  executes event  $S$ ; otherwise, it executes event  $\bar{S}$ . Intuitively,  $Q$  preserves the secrecy of  $x$  when the adversary cannot guess  $b$ , that is, it cannot distinguish whether the process outputs the value of the secret ( $b = \text{true}$ ) or outputs independent random numbers ( $b = \text{false}$ ).

**Correspondences** Correspondences [37] are properties of executed sequences of events, such as “if some event has been executed, then some other event has been executed”. They are typically used for formalizing authentication. Given a correspondence  $\text{corr}$ , we define a distinguisher  $D$  such that  $D(\mathcal{E}, a) = \text{true}$  if and only if the sequence of events  $\mathcal{E}$  satisfies the correspondence  $\text{corr}$ . We write this distinguisher simply  $\text{corr}$ , and write  $\neg \text{corr}$  for its negation.

**Definition 3** (Correspondence). *The process  $Q$  satisfies the correspondence  $\text{corr}$  with public variables  $V$  up to probability  $p$  if and only if, for all evaluation contexts  $C$  acceptable for  $Q$  with public variables  $V$  that do not contain events used in  $\text{corr}$ ,  $\Pr[C[Q] : \neg \text{corr}] \leq p(C)$ .*

We refer the reader to [11] for more details on the verification of correspondences in CryptoVerif. We have:

**Lemma 1.** *If  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$  and  $C$  is an acceptable evaluation context for  $Q$  with public variables  $V$ , then for all  $V' \subseteq V \cup \text{var}(C)$ ,  $C[Q]$  preserves the secrecy of  $x$  with public variables  $V'$  up to probability  $p'$  such that  $p'(C') = p(C'[C])$ .*

*If  $Q$  satisfies a correspondence  $\text{corr}$  with public variables  $V$  up to probability  $p$  and  $C$  is an acceptable evaluation context for  $Q$  with public variables  $V$  that does not contain events used in  $\text{corr}$ , then for all  $V' \subseteq V \cup \text{var}(C)$ ,  $C[Q]$  satisfies  $\text{corr}$  with public variables  $V'$  up to probability  $p'$  such that  $p'(C') = p(C'[C])$ .*

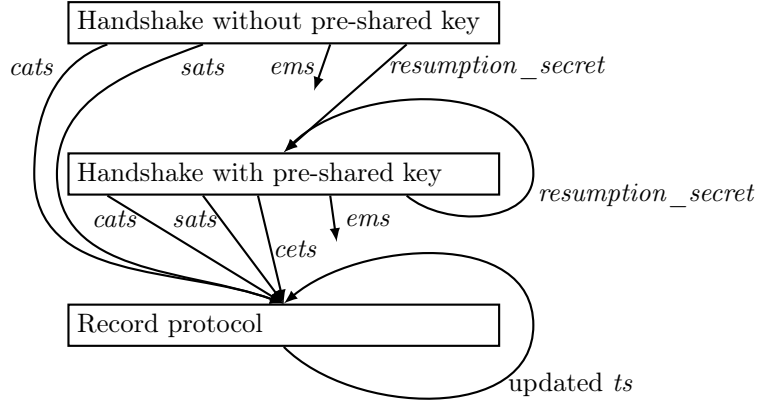


Figure 1: Structure of the composition

If  $Q \approx_p^{V \cup \{x\}} Q'$  and  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p'$ , then  $Q'$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p''$  such that  $p''(C) = p'(C) + 2 \times p(C[\cdot] \mid R_x, t_S)$ .

If  $Q \approx_p^V Q'$  and  $Q$  satisfies a correspondence  $\text{corr}$  with public variables  $V$  up to probability  $p'$ , then  $Q'$  satisfies  $\text{corr}$  with public variables  $V$  up to probability  $p''$  such that  $p''(C) = p'(C) + p(C, t_{\text{corr}})$ .

**Tables** CryptoVerif also supports tables. Tables are lists of tuples shared between all honest participants of the protocol. The construct **insert**  $Tbl(M_1, \dots, M_n); P$  inserts element  $(M_1, \dots, M_n)$  in table  $Tbl$ , then runs  $P$ . The construct **get**  $Tbl(x_1, \dots, x_l)$  **suchthat**  $M$  **in**  $P$  **else**  $P'$  tries to retrieve an element  $(x_1, \dots, x_l)$  in the table  $Tbl$  such that  $M$  is true. When such an element is found, it executes  $P$  with  $(x_1, \dots, x_l)$  bound to that element. When no such element is found, it executes  $P'$ . Equality tests  $= M_i$  are also allowed instead of variables  $x_i$ ; in this case, the table element must contain the value of  $M_i$  at the  $i$ -th position.

### 3 Structure of the proof of TLS 1.3

Our proof of TLS 1.3 relies on a previous analysis of the pieces that we compose [9, 10]. Figure 1 summarizes the structure of the composition. We provide a brief sketch of those previous results here; more details are given in Appendix B.1.

The initial handshake, without pre-shared key, provides 4 keys at the end of the protocol: the client traffic secret *cats*, used by the record protocol for messages from the client to the server; the server traffic secret *sats*, used by the record protocol for messages from the server to the client; the exporter master secret *ems*, used to compute exporters (secrets generated by TLS that can be used by applications or other protocols); and the resumption secret *resumption\_secret*, used as pre-shared key in the next handshake. For all these keys, CryptoVerif proves in particular secrecy, forward secrecy (with respect to the compromise of long-term client and server keys), and authentication.

In this model, the adversary has access to oracles that allow him to compromise the long-term client and server keys. The security properties are proved provided the long-term key of the peer is not compromised yet at the end of the handshake. As explained in the definition of secrecy (Section 2), this model does not include reveal queries for session keys; instead, CryptoVerif

proves that all keys of the various sessions are indistinguishable from independent random keys, which is a stronger model [1].

The handshake with pre-shared key uses a pre-shared key and provides the same keys as above, with the same security properties. Additionally, it provides a client early traffic secret *cets*, computed after the first message of the protocol (**ClientHello**). The record protocol uses this traffic secret to send messages from the client to the server immediately after the **ClientHello** message, so-called 0-RTT data. The **ClientHello** message may be replayed and the server may also accept an altered **ClientHello** message, so CryptoVerif proves weaker properties about *cets*. When the **ClientHello** message is not altered, it proves in particular secrecy and non-injective authentication, since replays are possible. When the **ClientHello** message is altered, it essentially proves that the server has a value of *cets* that no one else has. In this case, the goal is to show that the record protocol that uses this value of *cets* never accepts messages.

Due to limitations of CryptoVerif, we cannot prove forward secrecy with respect to the compromise of the pre-shared key in the case of a handshake with pre-shared key and Diffie-Hellman key exchange. Hence, all properties that we prove for the handshake with pre-shared key rely on the secrecy of the pre-shared key. In the analysis of this part of the protocol, we can then consider that the long-term signature keys of the client and the server are compromised, and let the adversary deal with certificates and signatures if they appear. Therefore, the only common secret between our models of the initial handshake and of the handshake with pre-shared key is the pre-shared key. Furthermore, the analysis of the initial handshake allows the compromise of these long-term signature keys at the end of the handshake, so the security properties that we prove for the initial handshake remain valid.

Finally, the record protocol uses a traffic secret to derive an updated traffic secret, used for key updates, and a key and an initialization vector, used for encrypting and decrypting messages with an authenticated encryption scheme. CryptoVerif proves secrecy of the updated traffic secret, injective message authentication, and message secrecy. (The adversary cannot distinguish which one of two sets of messages is encrypted, similarly to the property we mentioned for  $S_1^b$  in Example 1.) We also consider two variants of the record protocol for 0-RTT. In the first variant, the receiver is replicated, so we have non-injective message authentication instead of the injective one. This variant is useful to support replays of unaltered **ClientHello** messages. In the second variant, the sender is additionally removed, and we show that the receiver never accepts a message. This variant is useful for altered **ClientHello** messages. The only common secret between the handshakes and the record protocol is the traffic secret.

The goal of our case study is to combine all these results in order to obtain security results for the full TLS 1.3 protocol.

## 4 The Most Basic Composition Theorem

As a warm-up, we present a very simple composition theorem, explained below.

**Theorem 1.** *Let  $C$  be any context with one hole, without replications above the hole and without **event\_abort**. Let  $Q_1$  be a process without **event\_abort**. Let  $M$  be a term of type  $T$ . Let*

$$\begin{aligned} S_1 &= C[\text{let } k = M \text{ in } \overline{c_1}(); Q_1] \\ S_2 &= c_2(); \text{new } k : T; \overline{c_3}(); Q_2 \end{aligned}$$

*where  $c_1, c_2, c_3$  do not occur elsewhere in  $S_1, S_2$ ;  $k$  is the only variable common to  $S_1$  and  $S_2$ ;  $S_1$  and  $S_2$  have no common channel, no common event, and no common table; and  $k$  does not occur in  $C$  and  $Q_1$ . Let  $c'_1$  be a fresh channel. Let*

$$S_{\text{composed}} = C[\text{let } k = M \text{ in } \overline{c'_1}(); (Q_1 \mid Q_2)]$$

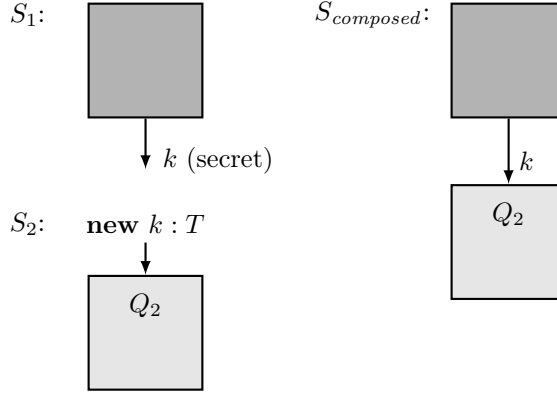


Figure 2: Illustration of Theorem 1

Let  $S_{composed}^\circ$  be obtained from  $S_{composed}$  by removing all events of  $S_1$ .

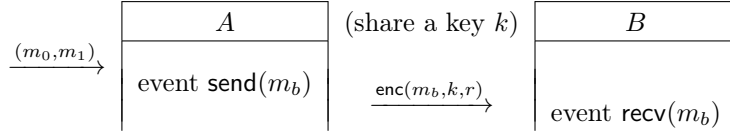
1. If  $S_1$  preserves the secrecy of  $k$  with public variables  $V$  ( $k \notin V$ ) up to probability  $p$ , then there exists an evaluation context  $C'$  such that, for any  $V_1 \subseteq V \cup (\text{var}(S_1) \setminus \{k\})$ , we have  $S_{composed}^\circ \approx_{p'}^{V_1} C'[S_2]$  and  $C'$  is acceptable for  $S_2$  without public variables, contains no event, runs in time at most  $t_C + t_{Q_1}$ , and does not alter the other parameters (replication bounds, lengths of bitstrings), where  $p'(C_1, t_D) = p(C'_1)$  and  $C'_1$  runs in time at most  $t_{C_1} + t_{Q_2} + t_D$  and its other parameters are the same as those of  $C_1$ .
2. There exists an evaluation context  $C''$  such that, for any  $V' \subseteq \text{var}(S_{composed})$ , we have  $S_{composed} \approx_0^{V'} C''[S_1]$  and  $C''$  is acceptable for  $S_1$  with public variable  $k$ , contains the events of  $S_2$ , runs in time at most  $t_{Q_2}$ , and does not alter the other parameters.

Moreover,  $C'$  is independent of  $Q_2$  and  $C''$  is independent of  $C$  and  $Q_1$ .

In this theorem, illustrated in Figure 2, we compose a system  $S_1$  that establishes a key  $k$  with a system  $S_2$  that runs  $Q_2$  using a fresh random key  $k$ . The composed system runs  $S_1$  and  $Q_2$  using the key  $k$  provided by  $S_1$ . (The letters  $Q$  and  $S$  both represent CryptoVerif processes, in the same grammar. We use  $S$  for the systems that we compose and for the composed system, and  $Q$  for other processes.) Intuitively, the composition works because the secrecy of  $k$  allows us to replace  $k$  with a fresh random key. (An adversary cannot distinguish  $k$  from a fresh random key.) A detailed proof is provided in Appendix A.4. In contrast to the theorems of Section 5, in this theorem,  $S_1$  is not a key exchange protocol: a single participant establishes the key  $k$ , so the composition is a lot easier.

The assumption that  $S_1$  does not contain **event\_abort** is useful because, in the definition of secrecy, when  $S_1$  aborts before a message is sent on  $c_s$ , neither  $S$  nor  $\bar{S}$  is executed, so the adversary gets no advantage against the secrecy of  $k$  for these traces. However, these traces could still leak information on  $k$  that would break the composition. So we prevent  $S_1$  from aborting. This is not a limitation in practice, because **event\_abort** is typically introduced during security proofs, using Shoup's lemma [13, 36], but does not occur in the initial protocol model.

The assumption that  $c_1, c_2, c_3$  do not occur elsewhere in  $S_1, S_2$  guarantees that messages sent to channel  $c_2$  (resp. received from  $c_1, c_3$ ) really go to the input (resp. come from the output) shown in the definitions of  $S_1$  and  $S_2$ . The assumption that  $k$  does not occur in  $C$  and  $Q$

Figure 3: A picture of system  $S_2^b$ 

guarantees that  $S_1$  defines  $k$  but does not use it. The other assumptions on  $S_1$  and  $S_2$  can easily be obtained by renaming if necessary.

The first conclusion of Theorem 1,  $S_{\text{composed}}^{\circ} \approx_{p'}^{V_1} C'[S_2]$ , allows us to transfer security properties from  $S_2$  to the composed system  $S_{\text{composed}}$  using Lemma 1. In this property, we need to remove the events of  $S_1$ , because events can leak information on  $k$  even when  $S_1$  preserves the secrecy of  $k$  according to Definition 2.

Similarly, the second conclusion of Theorem 1,  $S_{\text{composed}} \approx_0^{V'} C''[S_1]$ , allows us to transfer security properties from  $S_1$  to the composed system  $S_{\text{composed}}$ , provided these properties are proved with public variable  $k$ , because  $C''$  uses  $k$ . These properties may allow us to compose again  $S_{\text{composed}}$  with another protocol.

In our TLS case study, we use this composition theorem to deal with key updates in the record protocol. The system  $S_1$  runs the record protocol and computes an updated traffic secret from a traffic secret. This updated traffic secret is the key  $k$  in the composition theorem. The system  $S_2$  uses this key  $k$  to run the record protocol again. The composition theorem allows us to obtain security properties for a record protocol that performs a key update. We compose again recursively to allow any number of key updates. The next example presents a simplified version of this situation, to illustrate the theorem more formally.

**Example 1.** Consider the system  $S_2^b$  defined by

$$\begin{aligned}
 S_2^b = & c_2(); \mathbf{new } k : T; \overline{c_3} \langle \rangle; \\
 & (c_4((m_0 : T_m, m_1 : T_m)); \mathbf{new } r : T_r; \mathbf{event } \text{send}(m_b); \overline{c_5} \langle \text{enc}(m_b, k, r) \rangle) \\
 & | c_6(y : \text{bitstring}); \mathbf{let } i_{\perp}(m) = \text{dec}(y, k) \mathbf{ in event } \text{recv}(m)
 \end{aligned}$$

where all bitstrings in  $T_m$  have the same length. This system is illustrated in Figure 3. The system  $S_2^b$  chooses a key  $k$ , and then runs two participants, say  $A$  and  $B$ , in parallel. When  $A$  receives two messages  $m_0, m_1$  of the same length on channel  $c_4$ , it sends the encryption of  $m_b$  under  $k$  on channel  $c_5$  and records this emission with the event  $\text{send}(m_b)$ . When  $B$  receives a ciphertext on channel  $c_6$ , it decrypts that ciphertext, stores the plaintext in  $m$ , and executes event  $\text{recv}(m)$ . (The decryption function  $\text{dec}$  returns  $\perp$  when it fails, and the function  $i_{\perp}$  is the natural injection from  $\text{bitstring}$  to  $\text{bitstring} \cup \{\perp\}$ , so that the equality  $i_{\perp}(m) = \text{dec}(y, k)$  holds when the decryption succeeds and  $m$  is the corresponding cleartext.) When  $(\text{enc}, \text{dec})$  is an authenticated encryption scheme, we have  $S_2^0 \approx_{p_1} S_2^1$ , which means that the adversary can distinguish whether  $m_0$  or  $m_1$  was encrypted with probability at most  $p_1$ , and for  $b \in \{0, 1\}$ ,  $S_2^b$  satisfies the correspondence

$$\text{corr} = \mathbf{inj-event}(\text{recv}(m)) \implies \mathbf{inj-event}(\text{send}(m)) \quad (1)$$

up to probability  $p_2$  without public variables, which means that each execution of event  $\text{recv}(m)$  is preceded by a distinct execution of event  $\text{send}(m)$ , up to cases of probability at most  $p_2$ . (The probabilities  $p_1$  and  $p_2$  come from the probabilities of breaking the security properties of the

encryption scheme.) By composing  $S_1$  with  $S_2^b$ , we obtain

$$\begin{aligned} S_{composed}^b = & C[\text{let } k = M \text{ in } \overline{c_1}\langle \rangle; (Q_1 \\ & | c_4((m_0 : T_m, m_1 : T_m)); \text{new } r : T_r; \text{event send}(m_b); \overline{c_5}\langle \text{enc}(m_b, k, r) \rangle \\ & | c_6(y : \text{bitstring}); \text{let } i_\perp(m) = \text{dec}(y, k) \text{ in event recv}(m))] \end{aligned}$$

Let  $S_{composed}^{b,\circ}$  be obtained from  $S_{composed}^b$  by removing all events of  $S_1$ . Let  $V_1 = \text{var}(S_1) \setminus \{k\}$ . By Theorem 1, we have  $S_{composed}^{b,\circ} \approx_{p'}^{V_1} C'[S_2^b]$  for  $b \in \{0, 1\}$ . (The context  $C'$  does not depend on  $b$  because  $C'$  is independent of  $Q_2$  in Theorem 1.) By Lemma 1,  $C'[S_2^b]$  satisfies the correspondence (1) up to probability  $p'_2$  with public variables  $V_1$ , where  $p'_2(C_1) = p_2(C_1[C'])$ , and so  $S_{composed}^{b,\circ}$  satisfies (1) up to probability  $p'_2$ , where  $p'_2(C_1) = p'(C_1, t_{corr}) + p'_2(C_1) = p(C'_1) + p_2(C'_1)$ ,  $C'_1$  runs in time at most  $t_{C_1} + t_{enc} + t_{dec} + t_{corr}$ ,  $C'_1 = C_1[C']$  runs in time at most  $t_{C_1} + t_{C'} = t_{C_1} + t_C + t_{Q_1}$ , and their other parameters are the same as those of  $C_1$ . (The other parameters of  $C'_1 = C_1[C']$  are the same as those of  $C_1$  because  $C'$  does not alter these parameters.) Therefore,  $S_{composed}^b$  also satisfies (1) up to probability  $p'_2$ , since  $S_1$  does not contain the events `send` and `recv`. Moreover, assuming  $S_1$  does not contain events, we have  $S_{composed}^0 = S_{composed}^{0,\circ} \approx_{p'}^{V_1} C'[S_2^0] \approx_{p'_1}^{V_1} C'[S_2^1] \approx_{p'}^{V_1} S_{composed}^{1,\circ} = S_{composed}^1$  where  $p'_1(C_1, t_D) = p_1(C_1[C'], t_D)$ , so by transitivity,  $S_{composed}^0 \approx_{2p'+p'_1}^{V_1} S_{composed}^1$ : in the composed system, the adversary can distinguish whether  $m_0$  or  $m_1$  was encrypted with probability at most  $2p' + p'_1$ .

## 5 Main Composition Results

This section presents our main composition theorems. We first need to introduce preliminary notions and lemmas.

### 5.1 Transferring Security Properties

We first generalize the notion of indistinguishability. The more general notion still allows us to transfer security properties from a process to another, as indistinguishability does by Lemma 1.

**Definition 4.** We write  $Q \xrightarrow{f,p}^{V,V'} Q'$  if, and only if, for all evaluation contexts  $C$  acceptable for  $Q$  with public variables  $V$  and all distinguishers  $D$  that run in time at most  $t_D$ ,  $C' = f(C)$  is an evaluation context acceptable for  $Q'$  with public variables  $V'$  such that  $|\Pr[C[Q] : D] - \Pr[C'[Q'] : D]| \leq p(C, t_D)$ .

Intuitively,  $Q \xrightarrow{f,p}^{V,V'} Q'$  means that, for each adversary against  $Q$  (represented by the context  $C$ ), there exists a modified adversary against  $Q'$  (represented by the context  $C' = f(C)$ ) such that  $C[Q]$  and  $C'[Q']$  behave similarly. (The difference between the probabilities  $\Pr[C[Q] : D]$  and  $\Pr[C'[Q'] : D]$  is at most  $p(C, t_D)$ .)

Indistinguishability corresponds to the particular case in which  $f$  is the identity:  $f(C) = C$ . Being able to transform the context  $C$  by the function  $f$  is useful in composition proofs, in particular because the variables are not always numbered in the same way in the symmetric key protocol and in the composed system. In this case,  $f$  performs the renumbering of the variables.

The rest of this section shows that  $Q \xrightarrow{f,p}^{V,V'} Q'$  allows us to transfer indistinguishability, correspondence, and secrecy properties from  $Q'$  to  $Q$ .

**Lemma 2.** *If  $Q'_1 \approx_{p'}^{V'} Q'_2$ ,  $Q_1 \xrightarrow{f, p_1}^{V, V'} Q'_1$ , and  $Q_2 \xrightarrow{f, p_2}^{V, V'} Q'_2$ , then  $Q_1 \approx_{p''}^V Q_2$ , where  $p''(C, t_D) = p_1(C, t_D) + p'(f(C), t_D) + p_2(C, t_D)$ .*

Intuitively, if there is an adversary (represented by the context  $C$ ), that can distinguish  $Q_1$  from  $Q_2$  with probability  $p''$ , then the properties  $Q_1 \xrightarrow{f, p_1}^{V, V'} Q'_1$  and  $Q_2 \xrightarrow{f, p_2}^{V, V'} Q'_2$  guarantee that there is a modified adversary (represented by the context  $C' = f(C)$ ) that can distinguish  $Q'_1$  from  $Q'_2$  with probability at least  $p''(C, t_D) - p_1(C, t_D) - p_2(C, t_D)$ . Since  $Q'_1 \approx_{p'}^{V'} Q'_2$ , this probability is at most  $p'(f(C), t_D)$ , so we obtain Lemma 2. Lemma 3 is a similar result for correspondences.

**Lemma 3.** *If  $Q'$  satisfies a correspondence  $\text{corr}$  with public variables  $V'$  up to probability  $p'$  and  $Q \xrightarrow{f, p}^{V, V'} Q'$ , where  $f$  is such that when  $C$  does not contain events used by  $\text{corr}$ , neither does  $f(C)$ , then  $Q$  satisfies  $\text{corr}$  with public variables  $V$  up to probability  $p''$ , where  $p''(C) = p(C, t_{\text{corr}}) + p'(f(C))$ .*

**Definition 5.** *Assuming  $Q \xrightarrow{f, p}^{V, V'} Q'$ , we say that  $f$  is secrecy-preserving for  $x' \mapsto (x, f_{\text{sec}})$  when we have: If  $Q'$  preserves the secrecy of  $x'$  with public variables  $V' \setminus \{x'\}$  up to probability  $p'$ ,  $x' \in V'$ , and  $x \in V$ , then  $Q$  preserves the secrecy of  $x$  with public variables  $V \setminus \{x\}$  up to probability  $p''$ , where  $p''(C_0) = 2p(C_0[[ ] \mid R_x], t_S) + p'(f_{\text{sec}}(C_0))$ .*

Definition 5 just defines that function  $f$  allows us to transfer secrecy properties. This property holds in particular when, for every evaluation context  $C_0$  acceptable for  $Q \mid R_x$  with public variables  $V \setminus \{x\}$ , there exist  $C'_0$  and  $C''_0$  such that  $f(C_0[[ ] \mid R_x]) = C'_0[C''_0[[ ] \mid R_x]]$ . This condition guarantees that  $f$  preserves the form of contexts that we use to test secrecy  $C_0[[ ] \mid R_x]$ , just allowing the addition of a context  $C''_0$  before the secrecy test; this addition preserves secrecy by Lemma 1. (This result is detailed in Lemma 10 in Appendix A.5.) In our composition proofs, we use this condition, as well as others detailed in the proofs themselves.

## 5.2 Hash Oracles

The systems  $S_1$  and  $S_2$  that we compose may use hash oracles. In this paper, we consider only non-programmable random oracles. The systems  $S_1$  and  $S_2$  may share the same hash oracles, which appear once in the composed system. To allow the sharing of oracles between  $S_1$  and  $S_2$ , we must treat these oracles specially. In this section, we introduce notations and a lemma that allow us to do that.

We assume that there are  $L$  hash oracles ( $L \geq 1$ ), and use the following notations: for each  $l \leq L$ ,  $h_l$  is a function of type  $T_{hk_{h,l}} \times T_{h,l} \rightarrow T'_{h,l}$ ,

$$Q_h = \prod_{l=1}^L !^{i_{h,l} \leq n_{h,l}} c_{h3,l}[i_{h,l}](x_{h,l} : T_{h,l}); \overline{c_{h4,l}[i_{h,l}]}(h_l(hk_{h,l}, x_{h,l}))$$

$$C_h = c_{h1}(); \mathbf{new} \ hk_{h,1} : T_{hk_{h,1}}; \dots \mathbf{new} \ hk_{h,L} : T_{hk_{h,L}}; \overline{c_{h2}}(); ([ ] \mid Q_h)$$

The context  $C_h$  first chooses the keys  $hk_{h,l}$  ( $l \leq L$ ). This choice models the choice of the random oracles themselves. It is triggered by the reception of a message on  $c_{h1}$  and followed by an output on  $c_{h2}$ . Then,  $C_h$  runs the process  $Q_h$  in parallel with the hole. The process  $Q_h$  represents  $L$  hash oracles: the  $l$ -th hash oracle can be called at most  $n_{h,l}$  times; it receives its argument  $x_{h,l}$  on channel  $c_{h3,l}[i_{h,l}]$  ( $i_{h,l} \leq n_{h,l}$ ) and returns the hash of  $x_{h,l}$  on channel  $\overline{c_{h4,l}[i_{h,l}]}$ . We use  $\prod_{l=1}^L Q_l$  to denote the parallel composition  $Q_1 \mid \dots \mid Q_L$ . The context  $C_h$  is not an evaluation context (because it always chooses the keys  $hk_{h,l}$  before running the process in the hole). Let  $Q'_h$  and  $C'_h$  be obtained from  $Q_h$  and  $C_h$  by renaming the replication bounds  $n_{h,l}$  into  $n'_{h,l}$  and the channels



$c_{h1}, c_{h2}, c_{h3,l}, c_{h4,l}$  into  $c'_{h1}, c'_{h2}, c'_{h3,l}, c'_{h4,l}$  respectively. Similarly, let  $Q''_h$  and  $C''_h$  be obtained from  $Q_h$  and  $C_h$  by renaming the replication bounds  $n_{h,l}$  into  $n''_{h,l}$  and the channels  $c_{h1}, c_{h2}, c_{h3,l}, c_{h4,l}$  into  $c''_{h1}, c''_{h2}, c''_{h3,l}, c''_{h4,l}$  respectively. We say that a process is *hash-well-formed* when, for all  $l \leq L$ , it uses  $hk_{h,l}$  only in terms of the form  $h_l(hk_{h,l}, M)$  for some term  $M$ , it does not use the channels  $c_{h1}, c_{h2}, c_{h3,l}, c_{h4,l}, c'_{h1}, c'_{h2}, c'_{h3,l}, c'_{h4,l}, c''_{h1}, c''_{h2}, c''_{h3,l}, c''_{h4,l}$ , and it does not use the variables  $x_{h,l}$ .

In the particular case in which there is no hash oracle ( $L = 0$ ), we define  $C_h = C'_h = C''_h = []$ , the empty context.

Given a process  $Q$ , we write  $n_{h,l,Q}$  for the maximum number of evaluations of  $h_l(hk_{h,l}, \dots)$  in  $Q$ . The same notation applies to contexts  $C$  and terms  $M$ .

The next lemma is the main technical tool that we use to deal with hash oracles. It allows us to move the hash oracles under an evaluation context.

**Lemma 4.** *If  $C$  is an evaluation context and  $C[Q]$  is hash-well-formed, then there exists an evaluation context  $C'$  such that for all  $V$  such that  $V \cap \text{var}(C_h) = \emptyset$ ,*

$$C_h[C[Q]] \approx_0^V C'[C'_h[Q]]$$

where the context  $C'$  is independent of  $Q$ , runs in time at most  $t_C$ , and for all  $l \leq L$ ,  $C'$  calls the  $l$ -th hash oracle in  $C'_h$  at most  $n_{h,l,C}$  times, so  $n'_{h,l} = n_{h,l} + n_{h,l,C}$ . (The symbol  $n_{h,l}$  occur in  $C_h$  and  $n'_{h,l}$  occurs in  $C'_h$ .) The other parameters of  $C'$  are the same as those of  $C$ .

In this lemma, the context  $C$  directly calls the hash functions  $h_l$ , while the context  $C'$  performs the same hash evaluations by calling the hash oracles defined by  $Q'_h$  inside  $C'_h$ . (The context  $C'$  cannot call  $h_l$  directly, because it does not have access to the keys  $hk_{h,l}$ . The context  $C$  cannot call the hash oracles of  $C_h$  because it is hash-well-formed, so it does not use the channels  $c_{h3,l}$  and  $c_{h4,l}$ .)

### 5.3 Replication

When we compose a key exchange protocol  $S_1$  with a protocol  $S_2$  that uses the key, we typically run  $n$  sessions of the key exchange, and each session produces a fresh key. Therefore, we need to consider  $n$  independent sessions of  $S_2$ , each with a different fresh key. In this section, we show how to infer security properties (indistinguishability, secrecy, correspondences) of a protocol that runs  $n$  independent sessions from the properties of a protocol that runs a single session. (In the vocabulary of [16], we consider that the protocol that uses the key is *single-session reducible*, and we obtain results similar to theirs for the reduction to a single session [16, Appendix B], but in the context of CryptoVerif.)

Let us consider a protocol  $Q$  that runs a single session. We can model a protocol that runs  $n$  sessions of  $Q$  by adding a replication at the top of  $Q$ :  $!^{i \leq n} Q$ . Then all variables defined in  $Q$  implicitly have one more index,  $i$ , because they are defined under  $!^{i \leq n}$ . That allows us to distinguish the variables used in different sessions. However, this is not sufficient: we want the adversary to be able to know to (resp. from) which session it sends (resp. receives) messages, so we add the replication index  $i$  to the channels of inputs and outputs in  $Q$ . Similarly, we can add the replication index  $i$  as argument of events in  $Q$ , to be able to relate events that belong to the same session. Considering the process  $S_2^b$  of Example 1, that yields:

$$\begin{aligned} & !^{i \leq n} c_2[i](); \mathbf{new} \ k : T; \overline{c_3[i]} \langle \rangle; \\ & (c_4[i]((m_0 : T_m, m_1 : T_m)); \mathbf{new} \ r : T_r; \mathbf{event} \ \text{send}(i, m_b); \overline{c_5[i]} \langle \text{enc}(m_b, k, r) \rangle \\ & \mid c_6[i](y : \text{bitstring}); \mathbf{let} \ i_\perp(m) = \text{dec}(y, k) \ \mathbf{in} \ \mathbf{event} \ \text{recv}(i, m)) \end{aligned}$$

and this process satisfies the correspondence

$$\mathbf{inj\_event}(\mathbf{recv}(i, m)) \Longrightarrow \mathbf{inj\_event}(\mathbf{send}(i, m))$$

that is, each execution of  $\mathbf{recv}(i, m)$  is preceded by a distinct execution of  $\mathbf{send}(i, m)$ , up to cases of negligible probability. In this process, partnered sessions (which use the same key  $k$ ) have the same replication index  $i$ . However, this property is not preserved by composition: in a key exchange protocol, partnered sessions are typically the ones that exchange the same messages, and they do not necessarily have the same replication index. This will also be true in the composed system. Partnered sessions can then be determined by a *session identifier* computed from the messages exchanged in the protocol, as in [1, 7, 16, 19]: partnered sessions have the same session identifier. In the composition, the session identifier will be determined by the key exchange protocol. Therefore, we consider that the protocol that uses the key receives the session identifier in a variable  $x$ , as follows:

$$\begin{aligned} & !^{i \leq n} c_2[i](x : T_{\text{sid}}); \mathbf{new} \ k : T; \overline{c_3[i]} \langle \rangle; \\ & (c_4[i]((m_0 : T_m, m_1 : T_m)); \mathbf{new} \ r : T_r; \mathbf{event} \ \mathbf{send}(x, m_b); \overline{c_5[i]} \langle \mathbf{enc}(m_b, k, r) \rangle \\ & \mid c_6[i](y : \text{bitstring}); \mathbf{let} \ i_{\perp}(m) = \mathbf{dec}(y, k) \ \mathbf{in} \ \mathbf{event} \ \mathbf{recv}(x, m)) \end{aligned}$$

We use the session identifier  $x$  instead of the replication index  $i$  in events. The only missing ingredient in the above process is that the same session identifier should never be used twice, to avoid confusions between several sessions. The **find** construct allows us to verify that, by comparing  $x$  to the previously received session identifiers. This explanation leads us to the following definition:

**Definition 6.** Given a process  $P$ , and replication indices  $\tilde{i}$  and a variable  $x$  that do not occur in  $P$ , we write  $\mathbf{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, P)$  for the process obtained by adding indices  $\tilde{i}$  at the beginning of each sequence of indices of channels in inputs and outputs and at the beginning of the indices of each variable defined in  $P$  (implicit when current replication indices are omitted), adding variable  $x$  at the beginning of each event and at the beginning of each insertion in a table, and adding the test  $= x$  at the beginning of each **get** in a table.

Given a correspondence  $\text{corr}$ , we write  $\mathbf{AddSid}(T_{\text{sid}}, \text{corr})$  for the correspondence obtained by choosing a fresh variable  $x$  of type  $T_{\text{sid}}$  and adding it at the beginning of each event in  $\text{corr}$ .

When  $Q$  is of the form  $Q = c(); P$  and the channels  $c$  and  $c'$  and the replication indices  $\tilde{i}$  do not occur in  $P$ , we define

$$\begin{aligned} \mathbf{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q) &= !^{\tilde{i} \leq \tilde{n}} c'[\tilde{i}](x : T_{\text{sid}}); \\ \mathbf{find} \ \tilde{u} = \tilde{i}' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(x[\tilde{i}'], x'[\tilde{i}']) \wedge x &= x[\tilde{i}'] \ \mathbf{then} \ \mathbf{yield} \ \mathbf{else} \\ \mathbf{let} \ x' = \mathbf{cst} \ \mathbf{in} \ \mathbf{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, P) \end{aligned}$$

where  $x$ ,  $x'$ , and  $\tilde{u}$  are fresh variables.

The process  $\mathbf{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q)$  is the replicated version of process  $Q = c(); P$ : it corresponds to  $\tilde{n}$  copies of  $Q$  indexed by  $\tilde{i} \leq \tilde{n}$ , as shown by the replication  $!^{\tilde{i} \leq \tilde{n}}$ . However, it additionally manages the session identifier and replication indices as detailed in the explanation above. The first input in  $\mathbf{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q)$  receives the session identifier  $x$ , the subsequent **find** checks that the same  $x$  is never used twice, so that there is a bijection between the value of  $x$  and the replication indices  $\tilde{i}$ . (When the received session identifier  $x$  is equal to a previous one  $x[\tilde{i}']$  with which  $P$  was run, it just executes **yield**, which returns control to

the adversary. We record that  $P$  is run in session  $\tilde{i}$  by defining the variable  $x'[\tilde{i}]$  as a constant value `cst`. The **find** requires that  $x'[\tilde{i}']$  be defined, which means that  $P$  was run in session  $\tilde{i}'$ . In particular,  $\tilde{i}' \neq \tilde{i}$ , because  $x'[\tilde{i}]$  is not defined yet when the **find** is executed.) Finally, the process  $P$  that follows the input is executed, with the appropriate additions of the replication indices  $\tilde{i}$  or the session identifier  $x$  to channels, variables, events, and tables, as defined by  $\text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, P)$ .

Lemmas 5 and 6 below show that indistinguishability, secrecy, and correspondence properties are preserved by adding a replication. The hash oracles, when present, are left outside the replication.

**Lemma 5.** *Suppose that  $V \cap \text{var}(C'_h) = \emptyset$ ,  $Q = c(); P$ ,  $Q' = c(); P'$ ,  $Q$  and  $Q'$  are hash-well-formed and do not contain events,  $Q_! = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q)$ , and  $Q'_! = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q')$ . If  $C'_h[Q] \approx_p^V C'_h[Q']$ , then  $C_h[Q_!] \approx_{p'}^V C_h[Q'_!]$  where  $p'(C, t_D) = \prod \tilde{n} \times p(C', t_D)$  and the context  $C'$  runs in time at most  $t_C + (\prod \tilde{n} - 1) \times \max(t_Q, t_{Q'})$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n_{h,l} + (\prod \tilde{n} - 1) \times \max(n_{h,l,Q}, n_{h,l,Q'})$  times where  $C$  calls the  $l$ -th hash oracle at most  $n_{h,l}$  times, and the other parameters of  $C'$  are the same as those of  $C$ .*

**Lemma 6.** *Suppose that  $V \cap \text{var}(C'_h) = \emptyset$ ,  $Q$  is a hash-well-formed process, and  $Q_! = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\text{sid}}, Q)$ .*

1. *If  $C'_h[Q]$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$  with  $x \notin \text{var}(C'_h)$  and  $Q$  does not contain **event\_abort**, then  $C_h[Q_!]$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p'$ ; and*
2. *if  $C'_h[Q]$  satisfies the correspondence  $\text{corr}$  with public variables  $V$  up to probability  $p$ , then  $C_h[Q_!]$  satisfies the correspondence  $\text{AddSid}(T_{\text{sid}}, \text{corr})$  with public variables  $V$  up to probability  $p'$ ;*

where  $p'(C) = \prod \tilde{n} \times p(C')$  and the context  $C'$  runs in time at most  $t_C + (\prod \tilde{n} - 1)t_Q$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n_{h,l} + (\prod \tilde{n} - 1)n_{h,l,Q}$  times where  $C$  calls the  $l$ -th hash oracle at most  $n_{h,l}$  times, and the other parameters of  $C'$  are the same as those of  $C$ .

**Example 2.** Letting  $S_{2!}^b = \text{AddReplSid}(i \leq n, c'_2, T_{\text{sid}}, S_2^b)$ , by Lemma 5, we obtain  $S_{2!}^0 \approx_{p'_1} S_{2!}^1$  and by Lemma 6,  $S_{2!}^b$  satisfies the correspondence

$$\text{inj-event}(\text{recv}(x, m)) \implies \text{inj-event}(\text{send}(x, m))$$

up to probability  $p'_2$  without public variables, where  $p'_1(C, t_D) = n \times p_1(C', t_D)$ ,  $p'_2(C) = n \times p_2(C')$ , and  $C'$  runs in time at most  $t_C + (n - 1) \times t_{S_2^b}$  and its other parameters are the same as those of  $C$ . (Note that  $t_{S_2^0} = t_{S_2^1}$ .)

## 5.4 Main Composition Theorem

Finally, we obtain our main composition theorem. We write  $P\{M/x\}$  for the process obtained from  $P$  by substituting  $M$  for  $x$ . We denote by  $C + t_D$  a context that runs in time at most  $t_C + t_D$  and such that the other parameters of  $C + t_D$  are the same as those of  $C$ .

**Theorem 2** (Main composition theorem). *Let  $C$  be any context with two holes, with replications  $!\tilde{i} \leq \tilde{n}$  above the first hole and  $!\tilde{i}' \leq \tilde{n}'$  above the second hole and without **event\_abort**. Let  $Q_{1A}$  and  $Q_{1B}$  be processes without **event\_abort**. Let  $k, k_A, k_B$  be variables of type  $T$ . Let*

$$\begin{aligned} Q_1 = C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \text{let } k'_A = k_A \text{ in } \overline{c_A[\tilde{i}]}(M_A); Q_{1A}, \\ \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\tilde{i}']}(M_B); Q_{1B}] \end{aligned}$$

$$\begin{aligned}
Q_2 &= c_1(); \mathbf{new} \ k : T; \overline{c_2} \langle \rangle; (Q_{2A} \mid Q_{2B}) \\
S_1 &= C_h[Q_1] \\
S_2 &= C'_h[\text{AddReplSid}(\tilde{i} \leq \tilde{n}, c'_1, T_{\text{sid}}, Q_2)]
\end{aligned}$$

where  $Q_1$  and  $Q_2$  are hash-well-formed;  $\widetilde{msg}_A$  is a sequence of variables defined in  $C$  above the first hole and input or output by  $C$  above the first hole or by the output  $c_A[\tilde{i}] \langle M_A \rangle$ ;  $\widetilde{msg}_B$  is a sequence of variables input or output by  $C$  above the second hole;  $\text{sid}$  is a function that takes a sequence of messages and returns a session identifier of type  $T_{\text{sid}}$ ;  $C$ ,  $Q_{1A}$ ,  $Q_{1B}$ ,  $Q_{2A}$ , and  $Q_{2B}$  make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;  $c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$  do not occur elsewhere in  $S_1, S_2$ ;  $S_1$  and  $S_2$  have no common variable, no common channel, no common event, and no common table;  $S_1$  and  $S_2$  do not contain **newChannel**; and there is no **defined** condition in  $Q_2$ .

Let  $Q'_{2A} = \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_{2A})$  and  $Q'_{2B} = \text{AddIdxSid}(\tilde{i}' \leq \tilde{n}', x : T_{\text{sid}}, Q_{2B})$ . Let  $c'_A, c'_B$  be fresh channels. Let

$$\begin{aligned}
Q_{\text{composed}} &= \\
&C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \overline{c'_A[\tilde{i}]} \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \text{sid}(\widetilde{msg}_A)/x\}), \\
&\quad \mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\tilde{i}']} \langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \text{sid}(\widetilde{msg}_B)/x\})] \\
S_{\text{composed}} &= C''_h[Q_{\text{composed}}]
\end{aligned}$$

Let  $S_{\text{composed}}^\circ$  be obtained from  $S_{\text{composed}}$  by removing all events of  $S_1$ .

Let  $t_1 = t_C + \prod \tilde{n} \times (t_{M_A} + t_{Q_{1A}}) + \prod \tilde{n}' \times (t_{M_B} + t_{Q_{1B}})$  be an upper bound on the runtime of  $Q_1$ ,  $t_2 = \prod \tilde{n} \times t_{Q_{2A}} + \prod \tilde{n}' \times t_{Q_{2B}}$  be an upper bound on the runtime of  $Q'_{2A}$  and  $Q'_{2B}$  in  $Q_{\text{composed}}$ ,  $n_{h,l,1} = n_{h,l,C} + \prod \tilde{n} \times (n_{h,l,M_A} + n_{h,l,Q_{1A}}) + \prod \tilde{n}' \times (n_{h,l,M_B} + n_{h,l,Q_{1B}})$ , and  $n_{h,l,2} = \prod \tilde{n} \times n_{h,l,Q_{2A}} + \prod \tilde{n}' \times n_{h,l,Q_{2B}}$ .

1. If  $S_1$  preserves the secrecy of  $k'_A$  with public variables  $V$  ( $V \subseteq \text{var}(S_1) \setminus (\{k_A, k'_A\} \cup \text{var}(C_h))$ ) up to probability  $p$  and satisfies the correspondences

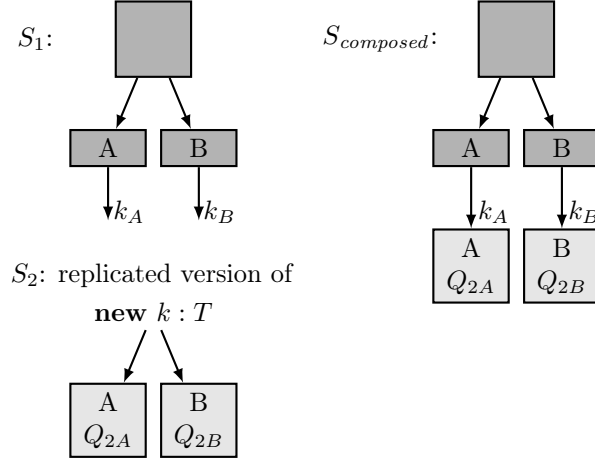
$$\mathbf{inj-event}(e_B(\text{sid}, k)) \implies \mathbf{inj-event}(e_A(\text{sid}, k, \tilde{i})) \quad (2)$$

$$\mathbf{event}(e_A(\text{sid}, k_1, \tilde{i}_1)) \wedge \mathbf{event}(e_A(\text{sid}, k_2, \tilde{i}_2)) \implies \tilde{i}_1 = \tilde{i}_2 \quad (3)$$

with public variables  $V \cup \{k'_A\}$  up to probabilities  $p'$  and  $p''$  respectively, then there exists  $f$  such that, for any  $V_1 \subseteq V \cup (\text{var}(Q_2) \setminus (\{k\} \cup \text{var}(C'_h)))$ , we have  $S_{\text{composed}}^\circ \xrightarrow{f, p_3}^{V_1, V_2} S_2$  where  $V_2 = V_1 \cap \text{var}(Q_2)$ ;  $p_3(C_3, t_D) = p(C'_3 + t_D) + p'(C'_3, t_D) + p''(C'_3, t_D)$  and, assuming  $C_3$  calls the  $l$ -th hash oracle  $n''_{h,l}$  times, the context  $C'_3$  runs in time at most  $t_{C_3} + t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l} = n''_{h,l} + n_{h,l,2}$  times, and its other parameters are the same as those of  $C_3$ ;  $f(C_3)$  contains the same events as  $C_3$ , runs in time at most  $t_{C_3} + t_1$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n''_{h,l} + n_{h,l,1}$  times, and its other parameters are the same as those of  $C_3$ ; if  $y \in V_2$ , then  $f$  is secrecy-preserving for  $y \mapsto (y, f_{\text{sec}})$  where  $f_{\text{sec}}(C_3)$  has the same parameters as  $f(C_3)$ .

2. There exists an evaluation context  $C'_4$  such that, for any  $V' \subseteq \text{var}(S_{\text{composed}}) \setminus (\{k'_A\} \cup \text{var}(C''_h))$ , we have  $S_{\text{composed}} \approx_0^{V'} C'_4[S_1]$  and  $C'_4$  is acceptable for  $S_1$  with public variables  $k'_A, k_B$ , contains the events of  $S_2$ , runs in time at most  $t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l,2}$  times, so  $n_{h,l} = n''_{h,l} + n_{h,l,2}$ , and does not alter the other parameters.

Moreover,  $f$  is independent of the details of  $Q_{2A}$  and  $Q_{2B}$ : it depends only on the channels of  $Q_{2B}$ , whether they are for input or for output, under which replications and with which type of data;  $C'_4$  is independent of  $Q_{1A}$  and  $Q_{1B}$ .



( $S_1$  may run several sessions of  $A$  and  $B$ .)

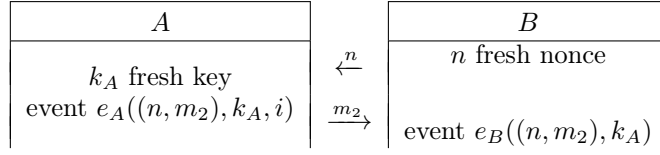
Figure 4: Illustration of Theorem 2

In this theorem, illustrated in Figure 4, the system  $S_1$  is a key exchange protocol that provides a key to two participants:  $A$  executes event  $e_A$  and stores the key in  $k_A$  and  $k'_A$ , and  $B$  executes event  $e_B$  and stores the key in  $k_B$ . The system  $S_2$  creates a fresh key, and also involves two participants:  $A$  executes  $Q_{2A}$  and  $B$  executes  $Q_{2B}$ . The composed system  $S_{composed}$  combines  $S_1$  and  $S_2$  so that  $A$  executes  $Q_{2A}$  with the key  $k_A$  and  $B$  executes  $Q_{2B}$  with the key  $k_B$ , after the key exchange  $S_1$  provides the key. These systems may share hash oracles, included in  $C_h$ ,  $C'_h$ , and  $C''_h$ . (These contexts use the same hash functions. The hash oracles are omitted in Figure 4.)

This theorem requires the key exchange to satisfy the following security properties. It must guarantee the secrecy of the key obtained by  $A$ ,  $k'_A$ , and injective authentication of  $A$  and  $B$ , as formalized by the correspondence (2). This correspondence means that each execution of event  $e_B(\widetilde{msg}, k)$  is preceded by a distinct execution of event  $e_A(\widetilde{msg}, k, \widetilde{i})$  for some  $\widetilde{i}$ , except in cases of probability at most  $p'$ . These two properties imply secrecy of the obtained key on  $B$ 's side, since all keys that  $B$  has are also keys that  $A$  has. The correspondence (3) means that the event  $e_A$  is executed at most once for each session identifier  $sid$ , since all such executions must have the same replication indices  $\widetilde{i}$ . It allows us to use the session identifier  $sid$  as argument  $x$  to identify the session in the system  $S_2$ . It is easy to prove in practice, both using CryptoVerif and manually: it is sufficient to notice that  $sid$  contains fresh randomness in each execution of  $e_A$ , for instance a nonce or an ephemeral public key.

The assumption that  $S_1$  and  $S_2$  do not contain **newChannel** guarantees that all channels are public. It is not a limitation in practice, because CryptoVerif does not support **newChannel** in protocol specifications; **newChannel** is used only in manual proofs. We require that the inputs and outputs use distinct channels with indices the current replication indices, to identify channels unambiguously. The assumption that there is no **defined** condition in  $Q_2$  facilitates a renumbering of variables: the variables of  $Q_{2B}$  have indices  $\widetilde{i}$  in  $S_2$  but  $\widetilde{i}'$  in  $S_{composed}$ . It is not a strong limitation since most usages of **find** with **defined** conditions can also be encoded using tables, and tables are not affected by the renumbering of variables.

Like Theorem 1, the first conclusion of Theorem 2 allows us to transfer security properties proved on  $S_2$  to  $S_{composed}$ , this time by relying on Section 5.1. We cannot prove indistinguishability here, because the variables of  $Q_{2B}$  are renumbered as mentioned above: since these variables



where  $m_2 = \text{enc}(\text{concat}(k_A, n), k_{lt}, r')$ .

Figure 5: A simple key exchange protocol

may be public, the renumbering may affect the context as well. The second conclusion allows us to transfer security properties proved on  $S_1$  to  $S_{composed}$  by Lemma 1, provided they are proved with public variables including  $k'_A$  and  $k_B$ , since  $C'_4$  uses  $k'_A$  and  $k_B$ .

In our TLS case study, we apply this theorem to perform most compositions: the handshakes with the record protocol, using a traffic secret as common key, as well as the handshake with pre-shared key with itself and the initial handshake with the handshake with pre-shared key, using the pre-shared key as common key. However, this theorem does not apply for the client early traffic secret *cets*, because of the possibility of replays. (Theorem 3 deals with this case.) The next example illustrates the theorem on a simpler case.

**Example 3.** Let us suppose that there are no hash oracles and consider the following very simple key exchange protocol, also shown in Figure 5:

```

 $S_1 = c_7(); \text{new } k_{lt} : T; \overline{c_8}();$ 
 $((!^{i_A \leq n_A} c_9[i_A](n : T_{nonce}); \text{new } k_A : T; \text{new } r' : T; \text{let } m_2 = \text{enc}(\text{concat}(k_A, n), k_{lt}, r') \text{ in}$ 
   $\text{event } e_A((n, m_2), k_A, i_A); \text{let } k'_A = k_A \text{ in } \overline{c_A[i_A]}(m_2))$ 
 $|$ 
 $(!^{i_B \leq n_B} c_{10}[i_B](); \text{new } n : T_{nonce}; \overline{c_{11}[i_B]}(n);$ 
 $c_{12}[i_B](m_2 : \text{bitstring}); \text{let } i_{\perp}(\text{concat}(k_B, n)) = \text{dec}(m_2, k_{lt}) \text{ in}$ 
 $\text{event } e_B((n, m_2), k_B); \overline{c_B[i_B]}())$ 

```

After an input on channel  $c_7$ , this process generates a long-term key  $k_{lt}$  shared between  $A$  and  $B$ , returns control to the adversary by outputting on channel  $c_8$ , and runs the participants  $A$  and  $B$  in parallel. The participant  $B$  (at the bottom) is run at most  $n_B$  times. It waits for an input on channel  $c_{10}[i_B]$ , generates a fresh nonce  $n$  and sends it to  $A$  on channel  $c_{11}[i_B]$ . If the session runs normally, the adversary forwards this nonce to channel  $c_9[i_A]$  for some  $i_A$ , so that  $A$  receives it, generates a fresh key  $k_A$ , and computes the message  $m_2$  that is the encryption of  $k_A$  and  $n$  under  $k_{lt}$ . (The function **concat** is concatenation.) Then,  $A$  executes the event  $e_A$  to record that it accepts the key  $k_A$ , in a session of session identifier  $(n, m_2)$ . (In this example, the function **sid** is the pair.) It stores  $k_A$  in  $k'_A$  and sends message  $m_2$  on channel  $c_A[i_A]$ . If the session runs normally, the adversary forwards this message to channel  $c_{12}[i_B]$ , so that  $B$  receives it, decrypts it, and in case of success, executes event  $e_B$  to record that it terminates with key  $k_B = k_A$ , in a session of session identifier  $(n, m_2)$ .

Assuming that  $(\text{enc}, \text{dec})$  is an authenticated encryption scheme, CryptoVerif shows that  $S_1$  preserves the secrecy of  $k'_A$  up to probability  $p$  and satisfies (2) and (3) with public variables  $k'_A, k_B$  up to probabilities  $p'$  and  $p''$  respectively, which depend on the probability of breaking the encryption scheme.

We compose  $S_1$  with the system  $S_2^b$  of Example 2. The syntactic assumptions are easy to

check, and the composed system is

$$\begin{aligned}
S_{composed}^b = & c_7(); \mathbf{new} \ k_{lt} : T; \overline{c_8} \langle \rangle; \\
& ((!^{i_A \leq n_A} c_9[i_A](n : T_{nonce}); \mathbf{new} \ k_A : T; \mathbf{new} \ r' : T_r; \mathbf{let} \ m_2 = \text{enc}(\text{concat}(k_A, n), k_{lt}, r') \ \mathbf{in} \\
& \quad \mathbf{event} \ e_A((n, m_2), k_A, i_A); \overline{c'_A[i_A]} \langle m_2 \rangle; \\
& \quad c_4[i_A]((m_0 : T_m, m_1 : T_m)); \mathbf{new} \ r : T_r; \mathbf{event} \ \text{send}((n, m_2), m_b); \overline{c_5[i_A]} \langle \text{enc}(m_b, k_A, r) \rangle) \\
& | \\
& (!^{i_B \leq n_B} c_{10}[i_B](); \mathbf{new} \ n : T_{nonce}; \overline{c_{11}[i_B]} \langle n \rangle; \\
& \quad c_{12}[i_B](m_2 : \text{bitstring}); \mathbf{let} \ i_\perp(\text{concat}(k_B, =n)) = \text{dec}(m_2, k_{lt}) \ \mathbf{in} \\
& \quad \mathbf{event} \ e_B((n, m_2), k_B); \overline{c'_B[i_B]} \langle \rangle; \\
& \quad c_6[i_B](y : \text{bitstring}); \mathbf{let} \ m = \text{dec}(y, k_B) \ \mathbf{in} \ \mathbf{event} \ \text{rcv}((n, m_2), m)))
\end{aligned}$$

The composed protocol runs the key exchange as before, then it sends  $m_b$  encrypted, as in  $S_2$ . However, in  $A$ , it executes event **send** with session identifier  $(n, m_2)$  and encrypts with key  $k_A$ . In  $B$ , it executes event **rcv** with session identifier  $(n, m_2)$  and decrypts with key  $k_B$ . These values are provided by the key exchange protocol. (The processes  $Q_{1A}$  and  $Q_{1B}$  are the process 0 that does nothing, so we simply omit them.)

Let  $S_{composed}^{b, \circ}$  be obtained from  $S_{composed}^b$  by removing events  $e_A$  and  $e_B$ . Let  $t_1 = t_2 = n_A t_{\text{enc}} + n_B t_{\text{dec}}$ . By Theorem 2, item 1), for  $b \in \{0, 1\}$ , there exists  $f$  such that  $S_{composed}^{b, \circ} \xrightarrow{f, p_3} S_{2!}^b$  where  $p_3(C_3, t_D) = p(C'_3 + t_D) + p'(C'_3, t_D) + p''(C'_3, t_D)$ ,  $C'_3$  runs in time at most  $t_{C_3} + t_2$ ,  $f(C_3)$  runs in time at most  $t_{C_3} + t_1$ , and their other parameters are the same as those of  $C_3$ . Since  $f$  does not depend on the details of  $S_{2!}^b$ ,  $f$  does not depend on  $b$ . Since  $S_{2!}^0 \approx_{p'_1} S_{2!}^1$ , by Lemma 2,  $S_{composed}^{0, \circ} \approx_{p_4} S_{composed}^{1, \circ}$  where  $p_4(C, t_D) = 2p_3(C, t_D) + p'_1(f(C), t_D)$ . Since  $S_{2!}^b$  satisfies (1) up to probability  $p'_2$ , by Lemma 3,  $S_{composed}^{b, \circ}$  satisfies (1) up to probability  $p_5(C) = p_3(C, t_{\text{corr}}) + p'_2(f(C))$ , and so does  $S_{composed}^b$ .

By Theorem 2, item 2), there exist evaluation contexts  $C_4^b$  such that  $S_{composed}^b \approx_0 C_4^b[S_1]$  and  $C_4^b$  is acceptable for  $S_1$  with public variables  $k'_A, k_B$ , runs in time at most  $t_2$ , and does not alter the other parameters. Since  $S_1$  satisfies (2) and (3) with public variables  $k'_A, k_B$  up to probabilities  $p'$  and  $p''$  respectively, by Lemma 1,  $C_4^b[S_1]$  and  $S_{composed}^b$  satisfy (2) and (3) up to probabilities  $p'_1(C) = p'(C[C_4^b])$  and  $p''_1(C) = p''(C[C_4^b])$  respectively. Therefore, we transferred security properties from  $S_1$  and  $S_{2!}^b$  to the composed system.

The properties required on  $S_1$  are closely related to the security of a key exchange protocol as defined in CryptoVerif [11]. As in [11], we require secrecy of the key obtained by  $A$  and injective authentication of  $A$  to  $B$  (2). The security of a key exchange includes mutual authentication, which is not needed for the composition. The correspondence (3) does not appear in [11]. Combined with (2), it implies that sessions that share the same session identifier have the same key:

$$\mathbf{event}(e_A(sid, k_A, \tilde{i}_1)) \wedge \mathbf{event}(e_B(sid, k_B)) \implies k_A = k_B \quad (4)$$

a property included in the definition of a key exchange in CryptoVerif [11]. Indeed, if  $e_A(sid, k_A, \tilde{i}_1)$  and  $e_B(sid, k_B)$  are executed, then  $e_A(sid, k_B, \tilde{i}_2)$  is also executed for some  $\tilde{i}_2$  by (2), which implies  $\tilde{i}_1 = \tilde{i}_2$  by (3), so the two events  $e_A(sid, k_A, \tilde{i}_1)$  and  $e_A(sid, k_B, \tilde{i}_2)$  are actually the same event, so  $k_A = k_B$ . The converse is not true in general, because (2) and (4) put no constraints in case event  $e_B$  is not executed with the considered session identifier.

These properties are also closely related to the properties used in previous composition results [16, 27, 29]. These results require key secrecy, as well as partnering or match security,

which provides guarantees similar to (2) and (3). In particular, [16, 29] require a public session matching algorithm, that is, the adversary knows which sessions are partnered. We also have this property: sessions are partnered when they have the same session identifier, and the session identifier is computed from public messages  $\widetilde{msg}_A$  and  $\widetilde{msg}_B$  by the function  $\text{sid}$ . This property is relaxed in [27]: they allow to use keys of early stages (which are virtually revealed) in the session matching. In TLS 1.3, the handshake is encrypted, and the session matching should be done on the plaintext, so the handshake keys are indeed needed for the session matching. In the model we consider, instead of encrypting the handshake, the handshake keys are given to the adversary, so that it can encrypt and decrypt messages. The session matching can then be done with public data.

However, the required properties still differ from [16, 27, 29] in their presentation. We make explicit the distinction between the two participants of the protocol, and (3) requires that  $A$  executes at most one session with a given session identifier. By (2), we obtain that  $B$  also executes at most one session with a given identifier. In contrast, [16, 27, 29] require that there are at most two sessions with the same identifier, without distinguishing  $A$  and  $B$ . The correspondence (2) guarantees that these two sessions have the same key, which is also required by [16, 27, 29].

Our definition of the key exchange protocol  $S_1$  allows much flexibility. In contrast to [16, 27, 29], we do not assume that the key exchange protocol is a public-key protocol. In TLS 1.3, the handshake with pre-shared key indeed relies on a shared key, and may not need a long-term public key. We encode “corrupt” queries, used to corrupt long-term keys, for instance to model forward secrecy, inside the context  $C$ . That allows us to deal with protocols that satisfy forward secrecy or not, without explicit distinction, in contrast to what [16, 27, 29] do. That also allows us to support keys that are forward secret only from a certain stage, as well as temporary keys, used in several sessions but not leaked by “corrupt” queries because their lifetime is short, as in the multi-stage framework of [27, 29]. As in [27, 29], the key exchange may continue running after accepting a key: it may send messages  $M_A$  and  $M_B$  and execute  $Q_{1A}$  and  $Q_{1B}$ . We allow composition with keys that are established before the last key exchange message, provided they are not used in the key exchange protocol. However, we cannot perform test queries on a stage- $i$  key and still use it in the key exchange protocol; while [27, 29] allow that, they do not allow composition for such keys. (In their vocabulary, these keys are not *final*.) As in [27], the communication partner does not need to be known at the start of the protocol. Finally, we support key exchange protocols that guarantee mutual authentication, unilateral authentication, or no authentication, as [27, 29]. This point may seem counter-intuitive, since (2) requires unilateral authentication. However, the security properties are obviously proved only when the partner is honest. Therefore, the system  $S_1$  executes the events  $e_A$ ,  $e_B$  and stores the key in  $k'_A$  only when the partner is honest. (That can be tested using **find** or tables when the partner is not authenticated.) Then, the correspondence (2) holds, and we can apply Theorem 2. When the partner is dishonest, we simply leak the key. Since no security property is desired in this case, we can trivially compose with any protocol that uses this key. This situation appears in TLS, when the client is not authenticated. In this case, the server considers that its partner is honest when the Diffie-Hellman share it receives has been sent by the honest client. This condition replaces client authentication and allows CryptoVerif to prove (2).

We give a proof sketch of Theorem 2 here. The full proof appears in Appendix A.9.

*Proof sketch.* Let

$$\begin{aligned}
 G_1 = & C''_h[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \\
 & \text{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \text{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k[\widetilde{i}''']) \wedge \\
 & \text{sid}(\widetilde{msg}_A) = \text{sid}(\widetilde{msg}_A[\widetilde{i}''']) \text{ then yield else} \\
 & \text{new } k : T; \overline{c'_A}[\widetilde{i}] \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{\text{sid}(\widetilde{msg}_A)/x\}),
 \end{aligned}$$



**event**  $e_B(\text{sid}(\widetilde{msg}_B), k_B);$   
**find**  $\tilde{u} = \tilde{i}'' \leq \tilde{n}$  **suchthat**  $\text{defined}(\widetilde{msg}_A[\tilde{i}''], k[\tilde{i}'']) \wedge$   
 $\text{sid}(\widetilde{msg}_A[\tilde{i}'']) = \text{sid}(\widetilde{msg}_B) \wedge \text{fresh}(\tilde{i}'', \tilde{u})$  **then**  
 $\overline{c'_B[\tilde{i}']\langle M_B \rangle}; (Q_{1B} \mid Q'_{2B}\{k[\tilde{u}]/k, \text{sid}(\widetilde{msg}_B)/x\})$

where  $\text{fresh}(\tilde{i}'', \tilde{u}) = \text{find } \tilde{u}' = \tilde{i}''' \leq \tilde{n}' \text{ suchthat } \text{defined}(\tilde{u}[\tilde{i}''']) \wedge \tilde{u}[\tilde{i}'''] = \tilde{i}'' \text{ then false else true}$ . We have  $\text{fresh}(\tilde{i}'', \tilde{u})$  when  $\tilde{i}''$  was not used before, that is, it does not occur in the array  $\tilde{u}$ . The game  $G_1$  runs the key exchange protocol followed by the protocol that uses the key, much like  $S_{composed}$ . However, in the participant  $A$  (after event  $e_A$ ), it does not run the protocol that uses the key when the same session identifier has already been seen in a previous session (**find**  $\tilde{u}''$ ), and it generates a fresh key  $k$  instead of using the key provided by the key exchange protocol (**new**  $k : T$ ). In the participant  $B$  (after event  $e_B$ ), it gets the key that has been generated in  $A$  with the same session identifier (**find**  $\tilde{u}$ ), and requires that the key of a given session of  $A$  is reused at most once by  $B$  (condition  $\text{fresh}(\tilde{i}'', \tilde{u})$ ).

Let  $Q'_2 = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c'_1, T_{\text{sid}}, Q_2)$ , so that  $S_2 = C'_h[Q'_2]$ . The first step of the proof consists in showing that  $G_1 \xrightarrow[f',0]{V_1, V_1 \cup \{\tilde{u}\}} C''_h[C'_5[Q'_2]]$  for some evaluation context  $C'_5$ . This is proved by establishing a correspondence between the traces of these two games. In this step, we renumber the variables of  $Q'_{2B}$ , replacing indices  $\tilde{a} \leq \tilde{n}'$  in  $G_1$  with  $\tilde{u}[\tilde{a}] \leq \tilde{n}$  in  $C''_h[C'_5[Q'_2]]$ . (The condition  $\text{fresh}(\tilde{i}'', \tilde{u})$  guarantees that  $\tilde{u}$  never takes twice the same value, hence the function from  $\tilde{a}$  to  $\tilde{u}[\tilde{a}]$  is injective. We exclude **defined** conditions in  $Q_2$  to facilitate this renumbering.) Since some of these variables may be public, this renumbering may also affect the context around  $G_1$  and  $C''_h[C'_5[Q'_2]]$ . That is why these two games are generally not indistinguishable.

Let  $G_1^\circ$  (resp.  $C_5^\circ$ ) be obtained from  $G_1$  (resp.  $C_5$ ) by removing all events of  $S_1$ . By Appendix A.6, Lemma 11, we have

$$G_1^\circ \xrightarrow[f',0]{V_1, V_1 \cup \{\tilde{u}\}} C''_h[C_5^\circ[Q'_2]] \quad (5)$$

By Lemma 4, there exists an evaluation context  $C'_5$  such that

$$C''_h[C_5^\circ[Q'_2]] \approx_0^{V_1 \cup \{\tilde{u}\}} C'_5[C'_h[Q'_2]] = C'_5[S_2] \quad (6)$$

where the context  $C'_5$  runs in time at most  $t_{C_5^\circ} \leq t_1$ , calls the  $l$ -th hash oracle in  $C'_h$  at most  $n_{h,l,C_5^\circ} \leq n_{h,l,1}$  times, so  $n'_{h,l} = n''_{h,l} + n_{h,l,1}$ , and its other parameters are the same as those of  $C_5^\circ$ .

The proof that  $S_{composed}^\circ \xrightarrow[f,p_3]{V_1, V_2} S_2$  then proceeds in two main steps:

1. First, we write the process  $G_1$  above as an evaluation context around

$$\begin{aligned}
S'_1 = & C_h[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \text{new } k'_A : T; \overline{c'_A[\tilde{i}]\langle (k'_A, M_A) \rangle}; Q_{1A}, \\
& \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\tilde{i}']\langle M_B \rangle}; Q_{1B}]]
\end{aligned}$$

Let  $S''_1$  be obtained by replacing **new**  $k'_A : T$  with **let**  $k'_A = k_A$  **in** in  $S'_1$ . Let  $G_2$  be obtained by replacing **new**  $k : T$  with **let**  $k = k_A$  **in** in  $G_1$ . Let  $G_2^\circ$ ,  $S_1'^\circ$ , and  $S_1''^\circ$  be obtained from  $G_2$ ,  $S'_1$ , and  $S''_1$  respectively by removing all events of  $S_1$ . Since  $S_1$  preserves the secrecy of  $k'_A$  with public variables  $V$  ( $k_A, k'_A \notin V$ ),  $S_1'^\circ$  is indistinguishable from  $S_1''^\circ$  (Appendix A.3, Lemma 9), so  $G_1^\circ$  is indistinguishable from  $G_2^\circ$ .

2. Second, we write  $G_2$  as an evaluation context around  $S_1$ . Since  $S_1$  satisfies the correspondences (2) and (3), so does  $G_2$ , so we obtain that, up to a small probability, the **find**  $\tilde{u}''$  in  $G_2$  fails and the **find**  $\tilde{u}$  succeeds with  $k[\tilde{u}] = k_B$ . From that, we show that  $G_2$  is indistinguishable from  $S_{composed}$ , so  $G_2^\circ$  is indistinguishable from  $S_{composed}^\circ$  (Appendix A.6, Lemma 11).

We conclude that  $S_{composed}^\circ \xrightarrow[f, p_3]{V_1, V_2} S_2$  with  $f(C_3) = f'(C_3)[C'_5[]]$  by combining these results with (5) and (6).

The proof that  $S_{composed} \approx_0^{V'} C'_4[S_1]$  is easier. Since  $k'_A$  and  $k_B$  are public variables, the context  $C'_4$  can use them. Since  $\widetilde{msg}_A$  and  $\widetilde{msg}_B$  are sent on public channels, the context  $C'_4$  also has access to them. Therefore, it can execute  $Q'_{2A}\{k'_A/k, \text{sid}(\widetilde{msg}_A)/x\}$  and  $Q'_{2B}\{k_B/k, \text{sid}(\widetilde{msg}_B)/x\}$  as the composed system does.  $\square$

## 5.5 Non-injective Variant

The next theorem is a variant of Theorem 2 with non-injective authentication. In this case, the process  $Q_{2B}$  may be executed several times for each key. Previous work [16, 27, 29] did not consider this case.

**Theorem 3** (Non-injective variant). *The conclusion of Theorem 2 still holds with the following changes in the hypotheses:  $Q_2 = c_1(); \text{new } k : T; \overline{c_2}\langle \rangle; (Q_{2A} \mid !^{\tilde{i}' \leq \tilde{n}'} Q_{2B})$ ,  $Q'_{2B} = \text{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\text{sid}}, Q_{2B})$  where the notation  $\emptyset$  designates the empty sequence, and the correspondence*

$$\text{event}(e_B(\text{sid}, k)) \implies \text{event}(e_A(\text{sid}, k, \tilde{i})) \quad (7)$$

instead of (2).

In the theorem above, the system  $S_1$  satisfies non-injective authentication: the correspondence (7) means that for each execution of  $e_B(\text{sid}, k)$ , there is an execution of  $e_A(\text{sid}, k, \tilde{i})$ . However, event  $e_B(\text{sid}, k)$  can be executed several times for each execution of  $e_A(\text{sid}, k, \tilde{i})$ . To compensate for that, the process  $Q_{2B}$  in  $Q_2$ , inside the system  $S_2$ , is replicated: it is under the replication  $!^{\tilde{i}' \leq \tilde{n}'}$ , with the same indices as those above  $e_B$ , so that it can also be executed several times for each shared key  $k$ . In the construction of the composed system, in  $Q'_{2B}$ , we do not need to add replication indices to  $Q_{2B}$ , since  $Q_{2B}$  already contains the replication indices  $\tilde{i}' \leq \tilde{n}'$ , because  $Q_{2B}$  is under the replication  $!^{\tilde{i}' \leq \tilde{n}'}$ . Hence, the construction of  $Q'_{2B}$  from  $Q_{2B}$  just adds the session identifier  $x$ .

In our TLS case study, we use this theorem to compose the handshake with pre-shared key with the record protocol using the client early traffic secret *cets* as common key. This theorem is needed because, in case **ClientHello** messages are replayed, several sessions of the server may obtain the same client early traffic secret, so the handshake does not guarantee injective authentication.

## 6 Application to TLS 1.3

In this section, we sketch the application of our composition theorems in order to compose the protocol pieces of TLS 1.3 as outlined in Figure 1. More details are given in Appendix B.2. The composition theorems are generally easy to apply: their assumptions are either proved by CryptoVerif or syntactic and easy to verify, and the composed protocol is syntactically built from the two pieces that we compose. The TLS case study still presents two additional difficulties:

- We compose protocols recursively an arbitrary number of times, in case there are successive handshakes with pre-shared keys or key updates in the record protocol, so we perform proofs by induction.
- The secrecy of payload messages is expressed by the secrecy of a bit  $b$  in a process that sends message  $m_b$  encrypted. We translate that into an indistinguishability between the process that sends  $m_0$  and the one that sends  $m_1$  (as  $S_2^0 \approx_{p_1} S_2^1$  in Example 1). Then we perform compositions on these two processes and combine the obtained results in order to prove secrecy of messages for composed processes.

The length of the composition proof is mostly due to the number of compositions that we perform between the various protocol pieces and the number of properties that we prove about these protocols.

In the composition, we first compose the record protocol with itself recursively by Theorem 1, using the secrecy of the updated traffic secret, to show that the security properties of the record protocol are preserved by key updates. We obtain a model of the record protocol that performs at most  $m$  key updates, for any  $m$ . We perform similar compositions for the 0-RTT variants. We put these protocols under replication by Lemmas 5 and 6, to model several sessions of the record protocol with independent traffic secrets.

Second, we compose the handshake with pre-shared key with the record protocol, using secret keys *cats* and *sats*, by Theorem 2. We also compose them with secret key *cets*, using Theorem 3 and the first 0-RTT variant of the record protocol, mentioned in Section 3, when the **ClientHello** message is unaltered, and using Theorem 5 (shown in Appendix A.12) and the second 0-RTT variant when the **ClientHello** message is altered. We also compose the obtained process with itself recursively, using the resumption secret *resumption\_secret* as pre-shared key in the next handshake, by Theorem 2, and put it under replication by Lemmas 5 and 6. These compositions yield processes that perform at most  $l$  successive handshakes with pre-shared key and  $m$  key updates.

Third, we compose the initial handshake with the record protocol, using secret keys *cats* and *sats*, by Theorem 2. We also compose the initial handshake with the process that runs handshakes with pre-shared key, using the resumption secret *resumption\_secret* as pre-shared key, by Theorem 2.

In all these compositions, CryptoVerif proves all secrecy and correspondence properties required by the theorems. The composed protocol inherits security properties from the components we compose. Therefore, these compositions allow us to infer security properties of the TLS protocol from properties of the handshakes and the record protocol. In particular, we obtain message secrecy, message forward secrecy (with respect to the compromise of long-term client and server keys), and injective message authentication for non-0-RTT application messages in both directions. For 0-RTT messages, since the handshake does not prevent replays for *cets*, we obtain non-injective authentication instead of the injective one. The correspondence properties of the handshakes are inherited by the composition and we also obtain secrecy of the exported master secrets *ems* provided by the various handshakes.

## 7 Conclusion

This paper presents several composition theorems, to compose a protocol that provides a key (e.g., a key exchange protocol) with a protocol that uses this key. These theorems rely on the computational model of cryptography. They are expressed in the framework of the tool CryptoVerif, so they are easily applicable when each protocol to compose is proved secure by

CryptoVerif. They provide great flexibility. In particular, they allow the composed protocols to share hash oracles, and they support non-injective as well as injective authentication.

We apply these theorems to TLS 1.3. This is an important case study, which illustrates well the power of our results. It allows us to prove security for any number of successive handshakes and key updates, a result that would be out of scope of CryptoVerif without composition, because this tool does not support loops. However, our theorems are of much more general interest, and we expect them to be applied to other protocols in the future. For instance, they apply as soon as a key exchange protocol provides a key to a cleanly separated transport protocol, a situation desirable in the design of many protocols.

Our results are specific to the CryptoVerif tool. We see no obstacle to recasting them in the framework of other tools that perform proofs in the computational model, such as EasyCrypt [5, 6]. However, although the general approach would be the same, a lot of our work would probably have to be redone to adapt the result to the language and formalism of each new tool. The assumptions of our theorems are either proved by CryptoVerif or syntactic and easy to verify. If desired, it would not be difficult to automate their verification and the application of the theorems in CryptoVerif. However, automating the application to TLS 1.3 would be more complicated, due to the additional difficulties mentioned at the beginning of Section 6. An interesting future work would also be to prove composition results with a key exchange protocol that already uses the key, for instance for key confirmation, in the line of [15].

**Acknowledgements** This work was partly supported by ANR TECAP (decision number ANR-17-CE39-0004-03) and H2020 NEXTLEAP.

## References

- [1] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. *IEEE Proceedings Information Security*, 153(1):27–39, Mar. 2006.
- [2] M. Arapinis, V. Cheval, and S. Delaune. Verifying privacy-type properties in a modular way. In *CSF’12*, pages 95–109. IEEE, June 2012.
- [3] M. Arapinis, V. Cheval, and S. Delaune. Composing security protocols: from confidentiality to privacy. In R. Focardi and A. Myers, editors, *POST’15*, volume 9036 of *LNCS*, pages 324–343. Springer, Apr. 2015.
- [4] G. Barthe, J. M. Crespo, Y. Lakhnech, and B. Schmidt. Mind the gap: Modular machine-checked proofs of one-round key exchange protocols. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 689–718. Springer, Apr. 2015.
- [5] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub. EasyCrypt: A tutorial. In A. Aldini, J. Lopez, and F. Martinelli, editors, *Foundations of Security Analysis and Design VII*, volume 8604 of *LNCS*, pages 146–166. Springer, 2014.
- [6] G. Barthe, B. Grégoire, S. Héraud, and S. Z. Béguelin. Computer-aided security proofs for the working cryptographer. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 71–90. Springer, Aug. 2011.
- [7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Eurocrypt’00*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.

- [8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Eurocrypt'06*, volume 4004 of *LNCS*, pages 409–426. Springer, May 2006. Extended version available at <http://ia.cr/2004/331>.
- [9] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *SECP'17*, pages 483–503. IEEE, May 2017.
- [10] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. Research Report RR-9040, Inria, May 2017. Available at <https://hal.inria.fr/hal-01528752>. CryptoVerif scripts available at <https://github.com/Inria-Prosecco/reftls/tree/master/cv>.
- [11] B. Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *CSF'07*, pages 97–111. IEEE, July 2007. Extended version available as ePrint Report 2007/128, <http://ia.cr/2007/128>.
- [12] B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, Oct.–Dec. 2008.
- [13] B. Blanchet. Automatically verified mechanized proof of one-encryption key exchange. In *CSF'12*, pages 325–339. IEEE, June 2012.
- [14] B. Blanchet. CryptoVerif: A computationally-sound security protocol verifier. Available at <http://cryptoverif.inria.fr/cryptoverif.pdf>, 2017.
- [15] C. Brzuska, M. Fischlin, N. P. Smart, B. Warinschi, and S. C. Williams. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, Aug. 2013.
- [16] C. Brzuska, M. Fischlin, B. Warinschi, and S. Williams. Composability of Bellare-Rogaway key exchange protocol. In *CCS'11*, pages 51–62. ACM, 2011.
- [17] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS'01*, pages 136–145. IEEE, Oct. 2001. An updated version is available at Cryptology ePrint Archive, <http://ia.cr/2000/067>.
- [18] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In S. Halevi and T. Rabin, editors, *TCC'06*, volume 3876 of *LNCS*, pages 380–403. Springer, Mar. 2006. Extended version available at <http://ia.cr/2004/334>.
- [19] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Eurocrypt'01*, volume 2045 of *LNCS*, pages 453–474. Springer, May 2001. Long version at <https://ia.cr/2001/040>.
- [20] R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *Crypto'03*, volume 2729 of *LNCS*, pages 265–281. Springer, Aug. 2003.
- [21] Ș. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *CSF'10*, pages 322–336. IEEE, July 2010.
- [22] V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, Feb. 2009.

- [23] A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *ENTCS*, 172:311–358, Apr. 2007.
- [24] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *CSFW’06*, pages 321–334. IEEE, July 2006.
- [25] S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In R. Kannan and K. Narayan Kumar, editors, *FSTTCS’09*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180. Leibniz-Zentrum für Informatik, Dec. 2009.
- [26] S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *CSF’08*, pages 239–251. IEEE, June 2008.
- [27] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *CCS’15*, pages 1197–1210, 2015. Full version available at <https://ia.cr/2015/914>.
- [28] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081, 2016. <https://ia.cr/2016/081>.
- [29] M. Fischlin and F. Günther. Multi-stage key exchange and the case of Google’s QUIC protocol. In *CCS’14*, pages 1193–1204, 2014. Full version available at [http://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications\\_1/fischlin-guenther-ccs2014.pdf](http://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications_1/fischlin-guenther-ccs2014.pdf).
- [30] T. Groß and S. Mödersheim. Vertical protocol composition. In *CSF’11*, pages 235–250. IEEE, June 2011.
- [31] J. D. Guttman and F. J. T. Fábrega. Protocol independence through disjoint encryption. In *CSFW’00*, pages 24–34. IEEE, July 2000.
- [32] R. Küsters and M. Tuengerthal. Composition theorems without pre-established session identifiers. In *CCS’11*, pages 41–50. ACM, 2011.
- [33] S. Mödersheim and L. Viganò. Secure pseudonymous channels. In M. Backes and P. Ning, editors, *ESORICS’09*, volume 5789 of *LNCS*, pages 337–354. Springer, Sept. 2009.
- [34] S. Mödersheim and L. Viganò. Sufficient conditions for vertical composition of security protocols. In *AsiaCCS’14*, pages 435–446. ACM, June 2014.
- [35] E. Rescorla. The Transport Layer Security (TLS) protocol version 1.3, draft-ietf-tls-tls13-28. <https://tools.ietf.org/html/draft-ietf-tls-tls13-28>, Mar. 2018.
- [36] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, Nov. 2004. Available at <http://ia.cr/2004/332>.
- [37] T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *SC’93*, pages 178–194. IEEE, May 1993.

## Appendices

### A Proofs

#### A.1 Characterization of Secrecy

Lemma 7 below gives a characterization of secrecy, which was used as a definition in versions of CryptoVerif that did not include **event\_abort** [11, 12]. We use it as an intermediate step in our proofs.

**Lemma 7.** *Let  $Q$  be a process that does not contain **event\_abort**. Let  $x$  and  $V$  be such that  $x \notin V$ . Let*

$$\begin{aligned} R_x^0 &= !^{i_s \leq n_s} c_s[i_s](\tilde{u} \leq \tilde{n}); \text{if defined}(x[\tilde{u}]) \text{ then } \overline{c_s[i_s]}(x[\tilde{u}]) \\ R_x^1 &= !^{i_s \leq n_s} c_s[i_s](\tilde{u} \leq \tilde{n}); \text{if defined}(x[\tilde{u}]) \text{ then} \\ &\quad \text{find } u_{s1} = i_{s1} \leq n_s \text{ suchthat defined}(y[i_{s1}], \tilde{u}[i_{s1}]) \wedge \tilde{u}[i_{s1}] = \tilde{u} \\ &\quad \text{then } \overline{c_s[i_s]}(y[u_{s1}]) \\ &\quad \text{else new } y : T; \overline{c_s[i_s]}(y) \end{aligned}$$

where the channel  $c_s$  and the variables  $\tilde{u}, u_{s1}, y$  do not occur in  $Q$  and the variable  $x$  has type  $T$  and is defined under replications  $!^{\tilde{i} \leq \tilde{n}}$  in  $Q$ . Let  $Q^\circ$  be obtained from  $Q$  by removing all events.

If the process  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$ , then  $Q^\circ \mid R_x^0 \approx_p^V Q^\circ \mid R_x^1$ , where  $p'(C, t_D) = p(C + t_D)$  and the context  $C + t_D$  runs in time at most  $t_C + t_D$  and its other parameters are the same as those of  $C$ .

Conversely, if  $Q^\circ \mid R_x^0 \approx_p^V Q^\circ \mid R_x^1$ , then  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$ , where  $p(C) = p'(C, t_D)$  and the distinguisher  $D$  is true when a certain event has been executed.

The processes  $R_x^0$  and  $R_x^1$  allow the adversary to query the variable  $x$ : if the adversary sends indices  $\tilde{u}$  on channel  $c_s[i_s]$ , and  $x[\tilde{u}]$  is defined, then the process  $R_x^0$  replies with the value of  $x[\tilde{u}]$ ; instead, the process  $R_x^1$  replies with a random value. The **find** in  $R_x^1$  makes sure that, if the indices  $\tilde{u}$  have already been queried, the previous reply is sent; otherwise, a fresh random value  $y$  is chosen in the type  $T$  of  $x$  by **new**  $y : T$ , and sent as a reply. Lemma 7 says that  $x$  is secret with public variables  $V$  if and only if an adversary with access to variables  $V$  cannot distinguish between  $Q^\circ \mid R_x^0$  and  $Q^\circ \mid R_x^1$ , that is, it cannot distinguish between the real values of  $x$  and independent random values.

*Proof.* Let us first prove that, if  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$ , then  $Q^\circ \mid R_x^0 \approx_p^V Q^\circ \mid R_x^1$ . Let  $C$  be an evaluation context acceptable for  $Q^\circ \mid R_x^0$  and  $Q^\circ \mid R_x^1$  with public variables  $V$ . Let  $c_{s0}$  and  $c'_s$  be channels that  $C$  does not use. We define a context  $C'$  ( $C + t_D$  in the statement of the lemma) that behaves like  $C$  except that:

- $C'$  starts by outputting on channel  $c_{s0}$  and inputting on channel  $c_{s0}$ , then it starts running  $C$ .
- When  $C$  executes an event **event**  $e(M_1, \dots, M_l)$ ,  $C'$  collects the executed event in its global state.
- When  $C$  terminates,  $C'$  recovers the sequence  $\mathcal{E}$  of executed events from its global state, and sends  $D(\mathcal{E}, 0)$  on channel  $c'_s$ .

Such a context  $C'$  exists because it can be encoded as a probabilistic Turing machine adversary, which can itself be encoded as a context in CryptoVerif [14, Section 2.8].

By definition of secrecy (Definition 2), when  $b$  is true,  $C'[Q \mid R_x]$  behaves like  $C[Q^\circ \mid R_x^0]$  and executes event  $S$  if and only if  $\text{true}$  is sent on channel  $c'_s$ , that is  $D(\mathcal{E}, 0)$  is true. (It is important that  $Q$  never aborts, so that  $C'[Q \mid R_x]$  can be programmed never to abort as well, and always sends some message on  $c'_s$ .) So

$$\Pr[C[Q^\circ \mid R_x^0] : D] = \Pr[C'[Q \mid R_x] : S/b = \text{true}].$$

Similarly, when  $b$  is false,  $C'[Q \mid R_x]$  behaves like  $C[Q^\circ \mid R_x^1]$  and executes event  $\bar{S}$  if and only if  $\text{true}$  is sent on channel  $c'_s$ , so

$$\Pr[C[Q^\circ \mid R_x^1] : D] = \Pr[C'[Q \mid R_x] : \bar{S}/b = \text{false}].$$

Therefore,

$$\begin{aligned} & \Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D] \\ &= \Pr[C'[Q \mid R_x] : S/b = \text{true}] - \Pr[C'[Q \mid R_x] : \bar{S}/b = \text{false}] \\ &= \Pr[C'[Q \mid R_x] : S \wedge b = \text{true}] / \Pr[b = \text{true}] - \Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{false}] / \Pr[b = \text{false}] \\ &= 2 \cdot \Pr[C'[Q \mid R_x] : S \wedge b = \text{true}] - 2 \cdot \Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{false}] \\ &= \Pr[C'[Q \mid R_x] : S \wedge b = \text{true}] + 1/2 - \Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{true}] - \\ & \quad (\Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{false}] + 1/2 - \Pr[C'[Q \mid R_x] : S \wedge b = \text{false}]) \end{aligned}$$

noting that  $\Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{true}] + \Pr[C'[Q \mid R_x] : S \wedge b = \text{true}] = \Pr[b = \text{true}] = 1/2$  since  $S$  or  $\bar{S}$  is always executed in this particular game, and similarly for  $b = \text{false}$ . Then

$$\begin{aligned} & \Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D] \\ &= \Pr[C'[Q \mid R_x] : S \wedge b = \text{true}] - \Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{true}] - \\ & \quad \Pr[C'[Q \mid R_x] : \bar{S} \wedge b = \text{false}] + \Pr[C'[Q \mid R_x] : S \wedge b = \text{false}] \\ &= \Pr[C'[Q \mid R_x] : S] - \Pr[C'[Q \mid R_x] : \bar{S}] \\ &\leq p(C') = p'(C, t_D) \end{aligned}$$

In case  $\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D] < 0$ , we consider the distinguisher  $\neg D$  that returns the negation of  $D$  and obtain

$$\begin{aligned} & \Pr[C[Q^\circ \mid R_x^1] : D] - \Pr[C[Q^\circ \mid R_x^0] : D] = \Pr[C[Q^\circ \mid R_x^0] : \neg D] - \Pr[C[Q^\circ \mid R_x^1] : \neg D] \\ &\leq p'(C, t_D) \end{aligned}$$

so we conclude that

$$|\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D]| \leq p'(C, t_D)$$

hence  $Q^\circ \mid R_x^0 \approx_{p'} Q^\circ \mid R_x^1$ .

Conversely, let us prove that, if  $Q^\circ \mid R_x^0 \approx_{p'}^V Q^\circ \mid R_x^1$ , then  $Q$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p$ , where  $p(C) = p'(C, t_D)$ . Let  $C$  be an evaluation context acceptable for  $Q \mid R_x$  with public variables  $V$  that does not contain  $S$  nor  $\bar{S}$ . We define a context  $C'$  that behaves like  $C$  except that:

- When  $C$  sends a message on channel  $c_{s0}$ ,  $C'$  immediately replies on channel  $c_{s0}$ .



- When  $C$  sends a bit  $b'$  on channel  $c'_s$ , if the bit  $b'$  is true, then  $C'$  executes **event\_abort**  $e$ , else  $C'$  executes **event\_abort**  $e'$ .
- When  $C$  terminates (without having sent a bit on  $c'_s$ ),  $C'$  chooses a random bit  $b'$ ; if  $b'$  is true, then  $C'$  executes **event\_abort**  $e$ , else  $C'$  executes **event\_abort**  $e'$ .

Let  $D$  be the distinguisher such that  $D(\mathcal{E}, a)$  is true if and only if  $\mathcal{E}$  contains event  $e$ .

When  $b$  is true,  $C'[Q^\circ \mid R_x^0]$  behaves like  $C[Q \mid R_x]$  and executes event  $e$  if and only if  $b' = \text{true}$  is sent on channel  $c'_s$  (that is,  $b' = b$ , that is, event  $S$  is executed) or  $C[Q \mid R_x]$  terminates without having sent a bit on  $c'_s$  and the random bit  $b'$  is true. So

$$\Pr[C'[Q^\circ \mid R_x^0] : D] = \Pr[C[Q \mid R_x] : S/b = \text{true}] + 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \text{true}].$$

(We write “term.” for “terminates without having sent a bit on  $c'_s$ .”) Similarly, when  $b$  is false,  $C'[Q^\circ \mid R_x^1]$  behaves like  $C[Q \mid R_x]$  and executes event  $e$  if and only if  $b' = \text{true}$  is sent on channel  $c'_s$  (that is,  $b' \neq b$ , that is, event  $\bar{S}$  is executed) or  $C[Q \mid R_x]$  terminates without having sent a bit on  $c'_s$  and the random bit  $b'$  is true, so

$$\Pr[C'[Q^\circ \mid R_x^1] : D] = \Pr[C[Q \mid R_x] : \bar{S}/b = \text{false}] + 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \text{false}].$$

Therefore,

$$\begin{aligned} & \Pr[C'[Q^\circ \mid R_x^0] : D] - \Pr[C'[Q^\circ \mid R_x^1] : D] \\ &= \Pr[C[Q \mid R_x] : S/b = \text{true}] - \Pr[C[Q \mid R_x] : \bar{S}/b = \text{false}] + \\ & \quad 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \text{true}] - 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \text{false}] \\ &= \Pr[C[Q \mid R_x] : S \wedge b = \text{true}] / \Pr[b = \text{true}] - \Pr[C[Q \mid R_x] : \bar{S} \wedge b = \text{false}] / \Pr[b = \text{false}] + \\ & \quad 1/2 \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \text{true}] / \Pr[b = \text{true}] - \\ & \quad 1/2 \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \text{false}] / \Pr[b = \text{false}] \\ &= 2 \cdot \Pr[C[Q \mid R_x] : S \wedge b = \text{true}] - 2 \cdot \Pr[C[Q \mid R_x] : \bar{S} \wedge b = \text{false}] + \\ & \quad \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \text{true}] - \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \text{false}] \\ &= \Pr[C[Q \mid R_x] : S \wedge b = \text{true}] + 1/2 - \Pr[C[Q \mid R_x] : \bar{S} \wedge b = \text{true}] - \\ & \quad (\Pr[C[Q \mid R_x] : \bar{S} \wedge b = \text{false}] + 1/2 - \Pr[C[Q \mid R_x] : S \wedge b = \text{false}]) \end{aligned}$$

noting that  $\Pr[C[Q \mid R_x] : \bar{S} \wedge b = \text{true}] + \Pr[C[Q \mid R_x] : S \wedge b = \text{true}] + \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \text{true}] = \Pr[b = \text{true}] = 1/2$  and similarly for  $b = \text{false}$ . So

$$\Pr[C'[Q^\circ \mid R_x^0] : D] - \Pr[C'[Q^\circ \mid R_x^1] : D] = \Pr[C[Q \mid R_x] : S] - \Pr[C[Q \mid R_x] : \bar{S}]$$

Hence

$$\begin{aligned} & \Pr[C[Q \mid R_x] : S] - \Pr[C[Q \mid R_x] : \bar{S}] = \Pr[C'[Q^\circ \mid R_x^0] : D] - \Pr[C'[Q^\circ \mid R_x^1] : D] \\ & \leq p'(C', t_D) = p(C') \end{aligned}$$

All parameters (runtime, replication bounds, lengths of bitstrings) are the same for  $C'$  and for  $C$  (up to very small runtimes) so we obtain  $\Pr[C[Q \mid R_x] : S] - \Pr[C[Q \mid R_x] : \bar{S}] \leq p(C)$ .  $\square$

## A.2 Eliminating Private Communications

In this section, we prove indistinguishability results by eliminating communications on private channels. Let us consider the following simple example. In the next lemma, we write  $P\{M/x\}$  for the process obtained from  $P$  by substituting  $M$  for  $x$ . Similarly, we will write  $Q_0\{\text{fr}_c/c, c \in S\}$  for the process obtained from  $Q_0$  by renaming the channel  $c$  into  $\text{fr}_c$  for each  $c \in S$ .

**Lemma 8.** *Let  $C$  be any context with replications  $!^{\tilde{i} \leq \tilde{n}}$  above the hole. Let  $P$  be a process, without **defined** conditions on the variable  $x$ . We have*

$$\mathbf{newChannel} \ c; (C[\overline{c[\tilde{i}]}(M)] \mid !^{\tilde{i} \leq \tilde{n}} c[\tilde{i}](x : T); P) \approx_0^V C[P\{M/x\}]$$

where the channel  $c$  does not occur elsewhere and  $V$  does not contain  $x$ .

The process on the left-hand side sends a message  $M$  on the private channel  $c[\tilde{i}]$ , and upon reception, the message is stored in  $x$  and  $P$  is executed. In the right-hand side, we shortcut the private communication and directly execute  $P$  with  $M$  substituted for  $x$ .

*Proof sketch.* The proof consists in relating the traces of the two processes in the presence of an evaluation context. It is easy to see that traces of the same probability match in the two processes, since the output on channel  $c[\tilde{a}]$  must be received by the input on  $c[\tilde{a}]$ . After that communication, the process  $P$  is then executed with the value of  $M$  for  $x$ . (Terms are deterministic, so evaluating  $M$  several times or none does not change the behavior of the processes.)  $\square$

We could prove other indistinguishability results obtained by eliminating private communications, using the same proof technique. Proving a general lemma would yield complex notations, so we prefer just referring to this proof technique when we need it in the following composition results.

### A.3 Consequence of Secrecy

The next lemma shows that, when a key  $k$  is secret, we can replace  $k$  with a fresh random key. The adversary cannot distinguish the real definition of  $k$  from the random one. In this lemma, we need to remove events, because events can leak information on  $k$  even when  $k$  is secret. That allows us to apply the previous characterization of secrecy (Lemma 7).

**Lemma 9.** *Let  $C$  be any context with replications  $!^{\tilde{i} \leq \tilde{n}}$  above the hole and without **event\_abort**. Let  $Q$  be a process without **event\_abort**. Let  $C^\circ$  and  $Q^\circ$  be obtained from  $C$  and  $Q$  respectively by removing all events. Let  $M$  be a term of type  $T$ . Suppose that  $k$  does not occur in  $C$ ,  $M$ ,  $M'$ , and  $Q$ . Let  $c_k, c'_k$  be channels that do not occur in  $C$  and  $Q$ . If  $C[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c_k[\tilde{i}]}(M'); Q]$  preserves the secrecy of  $k$  with public variables  $V$  ( $k \notin V$ ) up to probability  $p$ , then*

$$C^\circ[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c'_k[\tilde{i}]}((k, M')); Q^\circ] \approx_p^V C^\circ[\mathbf{new} \ k : T; \overline{c'_k[\tilde{i}]}((k, M')); Q^\circ]$$

where  $p'(C', t_D) = p(C' + t_D)$ .

*Proof.* Let  $G = C^\circ[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c_k[\tilde{i}]}(M'); Q^\circ]$ . Let  $c_s$  be a channel that does not occur in  $C$  and  $Q$ ,  $k'$  be a variable that does not occur in  $C$  and  $Q$ . By Lemma 7, since  $G$  does not contain events, we have  $G \mid R_x^0 \approx_p^V G \mid R_x^1$ . Suppose that  $M'$  has type  $T'$ . Let

$$C' = \mathbf{newChannel} \ c_k, c_s; \\ (!^{\tilde{i} \leq \tilde{n}} c_k[\tilde{i}](x : T'); \overline{c_s[\mathbf{encode}(\tilde{i})]}(\tilde{i}); c_s[\mathbf{encode}(\tilde{i})](k' : T); \overline{c'_k[\tilde{i}]}((k', x)) \mid [])$$

where  $\tilde{n} = n_1, \dots, n_m$  and  $\mathbf{encode}(\tilde{i})$  encodes the tuple  $\tilde{i} \in [1, n_1] \times \dots \times [1, n_m]$  as a element of  $[1, n_1 \times \dots \times n_m]$ .

We have

$$C'[G \mid R_x^0] \approx_0^V C^\circ[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c'_k[\tilde{i}]}((k, M')); Q^\circ]$$

by eliminating communications on  $c_k, c_s$  (Appendix A.2) and similarly

$$C'[G \mid R_x^1] \approx_0^V C^\circ[\mathbf{new} \ k : T; \overline{c'_k}[\tilde{i}] \langle (k, M') \rangle; Q^\circ]$$

since the variable  $k[\tilde{i}]$  is always defined when a message is sent on channel  $c_k[\tilde{i}]$ , so when  $R_x^1$  tests that definition, and furthermore, the indices  $\tilde{i}$  are sent at most once on channel  $c_s[\dots]$ , so  $R_x^1$  always replies with a fresh random element of type  $T$ . Moreover, since  $G \mid R_x^0 \approx_{p'}^V G \mid R_x^1$ , we have

$$C'[G \mid R_x^0] \approx_{p'}^V C'[G \mid R_x^1]$$

(we ignore the small runtime of  $C'$ ), so we obtain the desired result by transitivity of  $\approx$ .  $\square$

#### A.4 Proof for Section 4

*Proof of Theorem 1.* Let  $C^\circ$  and  $Q_1^\circ$  be obtained from  $C$  and  $Q_1$  respectively, by removing all events. Let  $C' = \mathbf{newChannel} \ c_2, c_3; (C^\circ[\overline{c_2} \langle \rangle; c_3 \langle \rangle; \overline{c'_1} \langle \rangle; Q_1^\circ] \mid [])$ . Let  $c'_1$  be a fresh channel. We have

$$\begin{aligned} C'[S_2] &\approx_0^{V_1} C^\circ[\mathbf{new} \ k : T; \overline{c'_1} \langle \rangle; (Q_1^\circ \mid Q_2)] \\ &\quad \text{by eliminating communications on channels } c_2 \text{ and } c_3 \text{ (Appendix A.2)} \\ &\approx_0^{V_1} \mathbf{newChannel} \ c'_1; (C^\circ[\mathbf{new} \ k : T; \overline{c'_1} \langle k \rangle; Q_1^\circ] \mid c'_1(k : T); \overline{c'_1} \langle \rangle; Q_2) \\ &\quad \text{by eliminating communications on channel } c'_1 \text{ (Appendix A.2)} \\ &\approx_{p'}^{V_1} \mathbf{newChannel} \ c'_1; (C^\circ[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c'_1} \langle k \rangle; Q_1^\circ] \mid c'_1(k : T); \overline{c'_1} \langle \rangle; Q_2) \\ &\quad \text{by Lemma 9 (We ignore the runtime of communication on } c'_1.) \\ &\approx_0^{V_1} S_{composed}^\circ \quad \text{by eliminating communications on channel } c'_1 \text{ (Appendix A.2)} \end{aligned}$$

Let  $k'$  be a fresh variable not in  $V'$ . Let

$$C'' = \mathbf{newChannel} \ c_1; (c_1 \langle \rangle; \mathbf{if} \ \mathbf{defined}(k) \ \mathbf{then} \ \mathbf{let} \ k' = k \ \mathbf{in} \ \overline{c'_1} \langle \rangle; Q_2\{k'/k\} \mid [])$$

We have

$$\begin{aligned} C''[S_1] &\approx_0^{V'} C[\mathbf{let} \ k = M \ \mathbf{in} \ \mathbf{if} \ \mathbf{defined}(k) \ \mathbf{then} \ \mathbf{let} \ k' = k \ \mathbf{in} \ \overline{c'_1} \langle \rangle; (Q_1 \mid Q_2\{k'/k\})] \\ &\quad \text{by eliminating communications on channel } c_1 \text{ (Appendix A.2)} \\ &\approx_0^{V'} C[\mathbf{let} \ k = M \ \mathbf{in} \ \overline{c'_1} \langle \rangle; (Q_1 \mid Q_2)] = S_{composed} \end{aligned}$$

since the condition  $\mathbf{defined}(k)$  is always true and  $k' = k$ .  $\square$

#### A.5 Proofs for Section 5.1

*Proof of Lemma 2.* Let  $C$  be any evaluation context acceptable for  $Q_1$  and  $Q_2$  with public variables  $V$ , and  $D$  be any distinguisher that runs in time at most  $t_D$ . Let  $C' = f(C)$ . The context  $C$  is an evaluation context acceptable for  $Q'_1$  and  $Q'_2$  with public variables  $V'$ . We have

$$\begin{aligned} &|\Pr[C[Q_1] : D] - \Pr[C[Q_2] : D]| \\ &\leq |\Pr[C[Q_1] : D] - \Pr[C'[Q'_1] : D]| + |\Pr[C'[Q'_1] : D] - \Pr[C'[Q'_2] : D]| + \\ &\quad |\Pr[C'[Q'_2] : D] - \Pr[C[Q_2] : D]| \\ &\leq p_1(C, t_D) + p'(f(C), t_D) + p_2(C, t_D) \\ &\leq p''(C, t_D) \end{aligned}$$

$\square$

*Proof of Lemma 3.* Let  $C$  be any evaluation context acceptable for  $Q$  with public variables  $V$  that does not contain events used in  $corr$ . Let  $C' = f(C)$ . The context  $C'$  is any evaluation context acceptable for  $Q'$  with public variables  $V'$  that does not contain events used in  $corr$ . We have

$$\begin{aligned} \Pr[C[Q] : \neg corr] &\leq |\Pr[C[Q] : \neg corr] - \Pr[C'[Q'] : \neg corr]| + \Pr[C'[Q'] : \neg corr] \\ &\leq p(C, t_{corr}) + p'(f(C)) = p''(C) \end{aligned}$$

□

**Lemma 10.** *If  $Q \xrightarrow{f,p}^{V,V'} Q'$ ,  $x \in V$ ,  $x' \in V'$ ,  $V \cap \text{var}(R_x) = \{x\}$ , and for every  $C_0$  evaluation context acceptable for  $Q \mid R_x$  with public variables  $V \setminus \{x\}$ , there exist  $C'_0$  and  $C''_0$  such that  $f(C_0[[\ ] \mid R_x]) = C'_0[C''_0[[\ ] \mid R_{x'}]]$ ,  $(\text{var}(C'_0) \cup \text{var}(C''_0)) \cap \text{var}(R_{x'}) = \emptyset$ ,  $\text{vardef}(C'_0) \cap \text{var}(C''_0) = \emptyset$ , and  $C'_0$  and  $C''_0$  do not use any common table, then  $f$  is secrecy-preserving for  $x' \mapsto (x, f_{\text{sec}})$  with  $f_{\text{sec}}(C_0) = C'_0[C''_0[[\ ] \mid R_{x'}]]$ .*

The set  $\text{vardef}(C'_0)$  contains the variables defined in  $C'_0$ .

*Proof.* Suppose  $Q \xrightarrow{f,p}^{V,V'} Q'$ ,  $x \in V$ ,  $x' \in V'$ , and  $Q'$  preserves the secrecy of  $x'$  with public variables  $V' \setminus \{x'\}$  up to probability  $p'$ . Let  $C_0$  be any evaluation context acceptable for  $Q \mid R_x$  with public variables  $V \setminus \{x\}$ . Let  $C'_0$  and  $C''_0$  be such that  $f(C_0[[\ ] \mid R_x]) = C'_0[C''_0[[\ ] \mid R_{x'}]]$ ,  $(\text{var}(C'_0) \cup \text{var}(C''_0)) \cap \text{var}(R_{x'}) = \emptyset$ ,  $\text{vardef}(C'_0) \cap \text{var}(C''_0) = \emptyset$ , and  $C'_0$  and  $C''_0$  do not use any common table. The context  $C'_0[C''_0[[\ ] \mid R_{x'}]]$  is an evaluation context acceptable for  $Q'$  with public variables  $V'$ . Since  $\text{var}(C''_0) \cap \text{var}(R_{x'}) = \emptyset$ ,  $C''_0$  does not use  $x'$ , so  $C''_0$  is an evaluation context acceptable for  $Q'$  with public variables  $V' \setminus \{x'\}$ , so by Lemma 1,  $C''_0[Q']$  preserves the secrecy of  $x'$  with public variables  $(V' \cup \text{var}(C''_0)) \setminus \{x'\}$  up to probability  $p_1$  such that  $p_1(C') = p'(C'[C''_0[Q']])$ . Since  $\text{var}(C'_0) \cap \text{var}(R_{x'}) = \emptyset$ ,  $C'_0$  does not use  $x'$ , so  $C'_0$  is an evaluation context acceptable for  $C''_0[Q'] \mid R_{x'}$  with public variables  $(V' \cup \text{var}(C''_0)) \setminus \{x'\}$ . To see that, please see the definition of acceptable evaluation contexts [14, Definition 4] and note that we have  $\text{var}(C'_0) \cap \text{var}(C''_0[Q'] \mid R_{x'}) \subseteq (\text{var}(C'_0) \cap \text{var}(C''_0)) \cup (\text{var}(C'_0) \cap \text{var}(Q')) \subseteq \text{var}(C'_0) \cup V'$ ,  $\text{vardef}(C'_0) \cap (\text{var}(C''_0) \cup V') = (\text{vardef}(C'_0) \cap \text{var}(C''_0)) \cup (\text{vardef}(C'_0) \cap V') = \emptyset$ , and  $C'_0$  and  $C''_0[Q'] \mid R_{x'}$  do not use any common table. We have

$$\begin{aligned} \Pr[C_0[Q \mid R_x] : S] - \Pr[C_0[Q \mid R_x] : \bar{S}] &\leq |\Pr[C_0[Q \mid R_x] : S] - \Pr[C'_0[C''_0[Q'] \mid R_{x'}] : S]| + \\ &\quad |\Pr[C'_0[C''_0[Q'] \mid R_{x'}] : S] - \Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \bar{S}]| + \\ &\quad |\Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \bar{S}] - \Pr[C_0[Q \mid R_x] : \bar{S}]| \\ &\leq p(C_0[[\ ] \mid R_x], t_S) + p_1(C'_0) + p(C_0[[\ ] \mid R_x], t_{\bar{S}}) \\ &\leq 2p(C_0[[\ ] \mid R_x], t_S) + p'(f_{\text{sec}}(C_0)) \end{aligned}$$

since  $t_S = t_{\bar{S}}$  and  $p_1(C'_0) = p'(C'_0[C''_0[[\ ] \mid R_{x'}]]) = p'(f_{\text{sec}}(C_0))$ . We conclude that  $Q$  preserves the secrecy of  $x$  with public variables  $V \setminus \{x\}$  up to probability  $p''$ , where  $p''(C_0) = 2p(C_0[[\ ] \mid R_x], t_S) + p'(f_{\text{sec}}(C_0))$ . Therefore,  $f$  is secrecy-preserving for  $x' \mapsto (x, f_{\text{sec}})$ . □

## A.6 Removing Events

As in the first conclusion of Theorem 1, we sometimes need to remove events. The next lemma shows that indistinguishability and the transfer relation of Section 5.1 are preserved by removing events.

**Lemma 11.** *Let  $Q_1^\circ$  and  $Q_2^\circ$  be obtained from  $Q_1$  and  $Q_2$  respectively by removing events  $e_1, \dots, e_n$ . If  $Q_1 \approx_p^V Q_2$ , then  $Q_1^\circ \approx_p^V Q_2^\circ$ . If  $Q_1 \xrightarrow{f,p}^{V,V'} Q_2$  and  $f$  commutes with renamings of events, then  $Q_1^\circ \xrightarrow{f,p}^{V,V'} Q_2^\circ$ .*

*Proof.* Let  $r(\mathcal{E})$  be the sequence of events obtained by removing events  $e_1, \dots, e_n$  from  $\mathcal{E}$ . Let  $\alpha$  be a renaming of  $e_1, \dots, e_n$  to fresh event names.

Let  $C$  be an evaluation context acceptable for  $Q_1^\circ$  and  $Q_2^\circ$  with public variables  $V$  and  $D$  be a distinguisher. The context  $\alpha(C)$  is also acceptable for  $Q_1^\circ$ ,  $Q_2^\circ$ ,  $Q_1$ , and  $Q_2$  with public variables  $V$ . We have

$$\begin{aligned} |\Pr[C[Q_1^\circ] : D] - \Pr[C[Q_2^\circ] : D]| &= |\Pr[\alpha(C)[Q_1^\circ] : D \circ \alpha^{-1}] - \Pr[\alpha(C)[Q_2^\circ] : D \circ \alpha^{-1}]| \\ &= |\Pr[\alpha(C)[Q_1] : D \circ \alpha^{-1} \circ r] - \Pr[\alpha(C)[Q_2] : D \circ \alpha^{-1} \circ r]| \\ &\leq p(C, t_D) \end{aligned}$$

neglecting the runtime of  $\alpha^{-1}$  and  $r$  and noticing that the renaming of events does not change the parameters of  $C$ , so  $Q_1^\circ \approx_p^V Q_2^\circ$ .

Let  $C$  be an evaluation context acceptable for  $Q_1^\circ$  with public variables  $V$  and  $D$  be a distinguisher. The context  $\alpha(C)$  is also acceptable for  $Q_1^\circ$  and for  $Q_1$  with public variables  $V$ , so letting  $C' = f(C)$ , the context  $\alpha(C') = \alpha(f(C)) = f(\alpha(C))$  is acceptable for  $Q_2$  and so for  $Q_2^\circ$  with public variables  $V'$ , so  $C'$  is also acceptable for  $Q_2^\circ$  with public variables  $V'$ . We have

$$\begin{aligned} |\Pr[C[Q_1^\circ] : D] - \Pr[C'[Q_2^\circ] : D]| &= |\Pr[\alpha(C)[Q_1^\circ] : D \circ \alpha^{-1}] - \Pr[\alpha(C')[Q_2^\circ] : D \circ \alpha^{-1}]| \\ &= |\Pr[\alpha(C)[Q_1] : D \circ \alpha^{-1} \circ r] - \Pr[\alpha(C')[Q_2] : D \circ \alpha^{-1} \circ r]| \\ &\leq p(C, t_D) \end{aligned}$$

so  $Q_1^\circ \xrightarrow{f,p}^{V,V'} Q_2^\circ$ . □

## A.7 Proof for Section 5.2

of Lemma 4. The lemma holds trivially when there is no random oracle: taking  $C' = C$ , we have  $C_h[C[Q]] \approx_0^V C'[C_h[Q]]$  because  $C_h[C[Q]] = C[Q] = C'[Q] = C'[C_h[Q]]$ . Let us now assume that there is at least one random oracle.

Suppose that  $C = \mathbf{newChannel} \tilde{c}; ([\mid Q_1])$ , with  $(\{c_{h1}, c_{h2}, c'_{h1}, c'_{h2}\} \cup \{c_{h3,l}, c_{h4,l}, c'_{h3,l}, c'_{h4,l} \mid l \leq L\}) \cap \{\tilde{c}\} = \emptyset$ . (We can generalize to any evaluation context by applying the result several times and by commuting parallel compositions if needed.)

Let  $Q'_1$  be obtained from  $Q_1$  by introducing assignments to fresh variables  $x_j$  ( $j \geq 1$ ) so that all occurrences of  $h_l(hk_{h,l}, M_j)$  are in processes  $\mathbf{let} \ x_j = h_l(hk_{h,l}, M_j) \ \mathbf{in}$ , and replacing these processes with

$$\overline{c'_{h3,l}[f_{l,j}(\tilde{i})]\langle M_j \rangle; c'_{h4,l}[f_{l,j}(\tilde{i})](x_j : T'_{h,l})}$$

where  $\tilde{i}$  are the replication indices above  $\mathbf{let} \ x_j = h_l(hk_{h,l}, M_j) \ \mathbf{in}$  and the functions  $f_{l,j}$  ( $j \geq 1$ ) and the functions  $f_{l,0}$  used below are chosen such that, for each  $l \leq L$ , the function  $(j, \tilde{i}) \mapsto f_{l,j}(\tilde{i})$  is injective.

Let  $y_{h,l}$  and  $y'_{h,l}$  be fresh variables. Let

$$\begin{aligned} C' &= \mathbf{newChannel} \ c'_{h1}, c'_{h2}, c'_{h3,l}, c'_{h4,l}, \tilde{c}; \\ &([\mid c_{h1}(); \overline{c'_{h1}}\langle \rangle; c'_{h2}(); \overline{c'_{h2}}\langle \rangle; Q'_1 \mid \\ &\prod_{l=1}^L !^{i_{h,l} \leq n_{h,l}} c_{h3,l}[i_{h,l}](y_{h,l} : T_{h,l}); \overline{c'_{h3,l}[f_0(i_{h,l})]\langle y_{h,l} \rangle; c'_{h4,l}[f_0(i_{h,l})](y'_{h,l} : T'_{h,l}); \overline{c_{h4,l}[i_{h,l}]\langle y'_{h,l} \rangle}) \end{aligned}$$

We have

$$\begin{aligned}
& C'[C'_h[Q]] \\
& \approx_0^V \mathbf{newChannel} \ c'_{h3,l}, c'_{h4,l}, \tilde{c}; \\
& \quad (c_{h1}(); \mathbf{new} \ hk_{h,1} : T_{hk_{h,1}}; \dots \mathbf{new} \ hk_{h,L} : T_{hk_{h,L}}; \overline{c_{h2}}\langle \rangle; (Q'_1 \mid Q \mid Q'_h) \mid \\
& \quad \prod_{l=1}^L \text{!}^{i_{h,l} \leq n_{h,l}} c_{h3,l}[i_{h,l}](y_{h,l} : T_{h,l}); \overline{c'_{h3,l}}[f_0(i_{h,l})]\langle y_{h,l} \rangle; \\
& \quad \quad c'_{h4,l}[f_0(i_{h,l})](y'_{h,l} : T'_{h,l}); \overline{c_{h4,l}}[i_{h,l}]\langle y'_{h,l} \rangle) \\
& \quad \text{by eliminating communications on } c'_{h1} \text{ and } c'_{h2} \text{ (Appendix A.2)} \\
& \approx_0^V \mathbf{newChannel} \ \tilde{c}; \\
& \quad (c_{h1}(); \mathbf{new} \ hk_{h,1} : T_{hk_{h,1}}; \dots \mathbf{new} \ hk_{h,L} : T_{hk_{h,L}}; \overline{c_{h2}}\langle \rangle; (Q_1 \mid Q \mid Q_h)) \\
& \quad \text{by eliminating communications on } c'_{h3,l} \text{ and } c'_{h4,l} \text{ for all } l \leq L \text{ (Appendix A.2)} \\
& \approx_0^V C_h[C[Q]]
\end{aligned}$$

In the second step, in case the adversary makes an output on  $c_{h3,l}$  before outputting on  $c_{h1}$  (and inputting on  $c_{h2}$ ), this output blocks immediately in the process after elimination of communications on  $c'_{h3,l}, c'_{h4,l}$  (because  $Q_h$  is not available yet); it succeeds in the process before elimination of communications on  $c'_{h3,l}, c'_{h4,l}$ , but the subsequent communication on  $c'_{h3,l}$  blocks (because  $Q'_h$  is not available yet). These two situations are indistinguishable.  $\square$

## A.8 Proofs for Section 5.3

We write  $\text{AddIdx}(\tilde{a}, Q)$  for the process obtained by adding indices  $\tilde{a}$  to all names of variables defined in  $Q$  and all names of events and tables in  $Q$ , and adding indices  $\tilde{a}$  at the beginning of all sequences of indices of channels in  $Q$ . This is just equivalent to renaming all variable, channel, event, and table names to distinct names for each value of  $\tilde{a}$ . Similarly, we write  $\text{AddIdx}(\tilde{a}, \text{corr})$  for the correspondence obtained by adding indices  $\tilde{a}$  to event names in  $\text{corr}$ .

*Proof of Lemma 5.* In this proof, we order the sequences  $\tilde{a}$  of the same length lexicographically and use  $\prod$  for the indexed parallel composition. Let  $Q_{\tilde{a}} = \text{AddIdx}(\tilde{a}, Q)$  and  $Q'_{\tilde{a}} = \text{AddIdx}(\tilde{a}, Q')$ . Let  $V_{\tilde{a}}$  be obtained by adding indices  $\tilde{a}$  to all variable names in  $V$ , and  $V' = \bigcup_{\tilde{a} \leq \tilde{n}} V_{\tilde{a}}$ . We have  $C'_h[Q_{\tilde{a}}] \approx_p^{V_{\tilde{a}}} C'_h[Q'_{\tilde{a}}]$ . By Lemma 4 applied with context  $C_{\tilde{a}} = (\prod_{\tilde{a}' < \tilde{a}} Q_{\tilde{a}'} \mid [] \mid (\prod_{\tilde{a}' > \tilde{a}, \tilde{a}' \leq \tilde{n}} Q'_{\tilde{a}'}))$ , there exists an evaluation context  $C'_{\tilde{a}}$  such that  $C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \approx_0^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]]$  and  $C_h[C_{\tilde{a}}[Q'_{\tilde{a}}]] \approx_0^{V'} C'_{\tilde{a}}[C'_h[Q'_{\tilde{a}}]]$ . Moreover, by adding context  $C'_{\tilde{a}}$ ,  $C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \approx_{p_{\tilde{a}}}^{V'} C'_{\tilde{a}}[C'_h[Q'_{\tilde{a}}]]$  where  $p_{\tilde{a}}(C_2, t_D) = p(C_2[C'_{\tilde{a}}], t_D)$ , so by transitivity,  $C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \approx_{p_{\tilde{a}}}^{V'} C_h[C_{\tilde{a}}[Q'_{\tilde{a}}]]$ . This result allows us to replace one process  $Q_{\tilde{a}}$  with  $Q'_{\tilde{a}}$ . We use a hybrid argument to replace all processes  $Q_{\tilde{a}}$  with  $Q'_{\tilde{a}}$  for  $\tilde{a} \leq \tilde{n}$ : by transitivity again,

$$C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \approx_{p'}^{V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q'_{\tilde{a}}]$$

where

$$p''(C_2, t_D) = \sum_{\tilde{a} \leq \tilde{n}} p_{\tilde{a}}(C_2, t_D) = \sum_{\tilde{a} \leq \tilde{n}} p(C_2[C'_{\tilde{a}}], t_D).$$

Let  $C$  be an acceptable evaluation context for  $C_h[Q!]$  and  $C_h[Q']$  with public variables  $V$ . We let  $C_1$  be obtained from  $C$  by renaming the variables  $y[\tilde{a}, \tilde{b}]$  into  $y_{\tilde{a}}[\tilde{b}]$  for  $y \in V$ . Let  $C_2 = C_1[\mathbf{newChannel} \ c; (!^{\tilde{i} \leq \tilde{n}} c'[\tilde{i}](x : T_{\text{sid}}); \mathbf{find} \ \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(x[\tilde{i}'], x'[\tilde{i}']) \wedge x = x[\tilde{i}'] \text{ then yield else let } x' = \text{cst in } c[\tilde{i}]\langle \rangle \mid [])]$ . We define  $f$  by  $f(C) = C_2$ .

We establish a correspondence between the traces of  $C[C_h[Q_!]]$  and the traces of  $C_2[C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]]$ : we eliminate communications on  $c$  and map variables  $y[\tilde{a}, \tilde{b}]$  to  $y_{\tilde{a}}[\tilde{b}]$  for  $y \in \text{var}(Q)$  and table entries  $Tbl(a', \tilde{b})$  to  $Tbl_{\tilde{a}}(\tilde{b})$  for tables  $Tbl$  of  $Q$  where  $\tilde{a}$  is the index such that  $x[\tilde{a}] = a'$  and  $x'[\tilde{a}]$  is defined. (The structure of the **find** in  $C_2$  guarantees that there exists exactly one such  $\tilde{a}$ :  $x'[\tilde{a}]$  is defined when the process is executed with  $x[\tilde{a}] = a'$  for the first time. The processes  $Q$  and  $Q'$  do not contain events, so events are not affected by the correspondence. The indices of channels are the same in both processes.) Therefore,  $\Pr[C[C_h[Q_!]] : D] = \Pr[C_2[C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]] : D]$ , so  $C_h[Q_!] \xrightarrow{f,0}^{V,V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]$ . Similarly,  $C_h[Q_!] \xrightarrow{f,0}^{V,V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q'_{\tilde{a}}]$ . From  $C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \approx_{p'''}^{V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q'_{\tilde{a}}]$  and these two properties, we conclude by Lemma 2 that

$$C_h[Q_!] \approx_{p'''}^{V'} C_h[Q'_!]$$

where  $p'''(C, t_D) = p''(C_2, t_D) = \sum_{\tilde{a} \leq \tilde{n}} p(C_2[C'_{\tilde{a}}], t_D)$ . Considering that the runtime of  $C_2$  is about the same as the runtime of  $C$ , the context  $C_2[C'_{\tilde{a}}]$  runs in time at most  $t_C + (\prod \tilde{n} - 1) \times \max(t_Q, t_{Q'})$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n_{h,l} + (\prod \tilde{n} - 1) \times \max(n_{h,l,Q}, n_{h,l,Q'})$  times, and its other parameters are the same as those of  $C$ . Therefore,  $p'''(C, t_D) \leq \prod \tilde{n} \times p(C', t_D) = p'(C, t_D)$ .  $\square$

*Proof of Lemma 6, Property 1.* Let  $Q^\circ$  and  $Q_!^\circ$  be obtained from  $Q$  and  $Q_!$  respectively by removing all events. By Lemma 7, we have  $C'_h[Q^\circ] \mid R_x^0 \approx_{p_1}^{V'} C'_h[Q^\circ] \mid R_x^1$  where  $p_1(C, t_D) = p(C + t_D)$ .

Let  $Q_{\tilde{a}} = \text{AddIdx}(\tilde{a}, Q^\circ)$ ,  $R_{x,\tilde{a}}^0 = \text{AddIdx}(\tilde{a}, R_x^0)$ , and  $R_{x,\tilde{a}}^1 = \text{AddIdx}(\tilde{a}, R_x^1)$ . Let  $V_{\tilde{a}}$  be obtained by adding indices  $\tilde{a}$  to all variable names in  $V$ , and  $V' = \bigcup_{\tilde{a} \leq \tilde{n}} V_{\tilde{a}}$ . We have

$$C'_h[Q_{\tilde{a}}] \mid R_{x,\tilde{a}}^0 \approx_{p_1}^{V_{\tilde{a}}} C'_h[Q_{\tilde{a}}] \mid R_{x,\tilde{a}}^1 \quad (8)$$

By Lemma 4 applied with the context  $C_{\tilde{a}} = [] \mid (\prod_{\tilde{a}' \leq \tilde{n}, \tilde{a}' \neq \tilde{a}} Q_{\tilde{a}'})$ , there exists an evaluation context  $C'_{\tilde{a}}$  such that  $C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \approx_0^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]]$ , so by adding context  $[] \mid R_{x,\tilde{a}}^0$ ,

$$\begin{aligned} C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^0 &\approx_0^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^0 \\ &\approx_0^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^1 \end{aligned}$$

since the channels restricted by  $C'_{\tilde{a}}$  do not occur in  $R_{x,\tilde{a}}^0$ , and similarly

$$C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^1 \approx_0^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^1$$

Moreover, by adding context  $C'_{\tilde{a}}$  to (8),

$$C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^0 \approx_{p_{\tilde{a}}}^{V'} C'_{\tilde{a}}[C'_h[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^1$$

where  $p_{\tilde{a}}(C_2, t_D) = p_1(C_2[C'_{\tilde{a}}], t_D)$ , so by transitivity,

$$C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^0 \approx_{p_{\tilde{a}}}^{V'} C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \mid R_{x,\tilde{a}}^1$$

Let  $C''_{\tilde{a}} = (\prod_{\tilde{a}' < \tilde{a}} R_{x,\tilde{a}'}^0) \mid [] \mid (\prod_{\tilde{a}' > \tilde{a}, \tilde{a}' \leq \tilde{n}} R_{x,\tilde{a}'}^1)$ . By adding context  $C''_{\tilde{a}}$ , we obtain

$$C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid C''_{\tilde{a}}[R_{x,\tilde{a}}^0] \approx_{p'_{\tilde{a}}}^{V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid C''_{\tilde{a}}[R_{x,\tilde{a}}^1]$$

where  $p'_{\tilde{a}}(C, t_D) = p_{\tilde{a}}(C[C''_{\tilde{a}}], t_D) = p_1(C[C''_{\tilde{a}}][C'_{\tilde{a}}], t_D) = p(C[C''_{\tilde{a}}][C'_{\tilde{a}}] + t_D)$ . By transitivity,

$$C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid (\prod_{\tilde{a} \leq \tilde{n}} R_{x,\tilde{a}}^0) \approx_{p'''}^{V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid (\prod_{\tilde{a} \leq \tilde{n}} R_{x,\tilde{a}}^1)$$

where  $p''(C, t_D) = \sum_{\tilde{a} \leq \tilde{n}} p'_a(C, t_D)$ .

Given a process  $Q$  and a replication index  $i$  that does not occur in  $Q$ , we write  $\text{AddIdx}_1(i \leq n, Q)$  for the process obtained by adding index  $i$  at the beginning of each sequence of indices of channels in inputs and outputs, at the beginning of each event, at the beginning of the indices of each variable defined in  $Q$  (implicit when current replication indices are omitted), and at the beginning of each insertion in a table, and adding the test  $= i$  at the beginning of each **get** in a table. We define  $\text{AddRepl}(i \leq n, Q) = !^{i \leq n} \text{AddIdx}_1(i \leq n, Q)$  and  $\text{AddRepl}(\tilde{i} \leq \tilde{n}, Q) = \text{AddRepl}(i_1 \leq n_1, \dots, \text{AddRepl}(i_m \leq n_m, Q))$  when  $\tilde{i} = i_1, \dots, i_m$  and  $\tilde{n} = n_1, \dots, n_m$ .

Using the same function  $f$  and trace correspondence as in Lemma 5, we show that

$$\begin{aligned} C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^0) &\xrightarrow{f, 0}^{V, V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid (\prod_{\tilde{a} \leq \tilde{n}} R_{x, \tilde{a}}^0) \\ C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^1) &\xrightarrow{f, 0}^{V, V'} C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}] \mid (\prod_{\tilde{a} \leq \tilde{n}} R_{x, \tilde{a}}^1) \end{aligned}$$

so by Lemma 2, we have

$$C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^0) \approx_{p'_1}^V C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^1) \quad (9)$$

where  $p'_1(C, t_D) = p''(f(C), t_D)$ .

Suppose that the variable  $x$  has type  $T$  and is defined under replications  $!^{\tilde{i}' \leq \tilde{n}'}$  in  $Q$ . Let  $c'_s$  be a fresh channel. Let

$$\begin{aligned} C_2 = & \text{newChannel } c_s; \\ & (!^{i'_s \leq n'_s} c'_s[i'_s](\tilde{u} \leq \tilde{n}, \tilde{u}' \leq \tilde{n}'); \overline{c_s}[\tilde{u}, i'_s](\tilde{u}'); c_s[\tilde{u}, i'_s](x' : T); \overline{c'_s}[\tilde{i}'_s](x') \mid []) \end{aligned}$$

Let

$$\begin{aligned} R_x'^0 = & !^{i'_s \leq n'_s} c'_s[i'_s](\tilde{u} \leq \tilde{n}, \tilde{u}' \leq \tilde{n}'); \text{if defined}(x[\tilde{u}, \tilde{u}']) \text{ then } \overline{c'_s}[\tilde{i}'_s](x[\tilde{u}, \tilde{u}']) \\ R_x'^1 = & !^{i'_s \leq n'_s} c'_s[i'_s](\tilde{u} \leq \tilde{n}, \tilde{u}' \leq \tilde{n}'); \text{if defined}(x[\tilde{u}, \tilde{u}']) \text{ then} \\ & \text{find } u'_{s1} = i'_{s1} \leq n'_{s1} \text{ suchthat defined}(y[i'_{s1}], \tilde{u}[i'_{s1}], \tilde{u}'[i'_{s1}]) \wedge \tilde{u}[i'_{s1}] = \tilde{u} \wedge \tilde{u}'[i'_{s1}] = \tilde{u}' \\ & \text{then } \overline{c'_s}[\tilde{i}'_s](y[u'_{s1}]) \\ & \text{else new } y : T; \overline{c'_s}[\tilde{i}'_s](y) \end{aligned}$$

be processes  $R_x^0$  and  $R_x^1$  associated to  $C_h[Q_!^\circ]$ , using channel  $c'_s$  instead of  $c_s$ . We have

$$\begin{aligned} C_2[C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^0)] &\approx_0^V C_h[Q_!^\circ] \mid R_x'^0 \\ C_2[C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^1)] &\approx_0^V C_h[Q_!^\circ] \mid R_x'^1 \end{aligned}$$

by eliminating communications on  $c_s$  (Appendix A.2). Moreover, by adding context  $C_2$  to (9), we obtain

$$C_2[C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^0)] \approx_{p'_1}^V C_2[C_h[Q_!^\circ] \mid \text{AddRepl}(\tilde{i} \leq \tilde{n}, R_x^1)]$$

(We ignore the very small runtime of  $C_2$ .) So by transitivity of  $\approx$ , we have

$$C_h[Q_!^\circ] \mid R_x'^0 \approx_{p'_1}^V C_h[Q_!^\circ] \mid R_x'^1$$

By Lemma 7, we conclude that  $C_h[Q_!]$  preserves the secrecy of  $x$  with public variables  $V$  up to probability  $p'$ , with  $p'(C) = p'_1(C, t_D) = p''(f(C), t_D) = \sum_{\tilde{a} \leq \tilde{n}} p'_a(f(C), t_D) =$



$\sum_{\tilde{a} \leq \tilde{n}} p(f(C)[C''_{\tilde{a}}[C'_{\tilde{a}}]] + t_D)$ . Since  $D$  is true when an event is executed, its runtime  $t_D$  can be neglected. Moreover, the context  $f(C)[C''_{\tilde{a}}[C'_{\tilde{a}}]]$  runs in time at most  $t_C + (\prod \tilde{n} - 1)t_Q$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n_{h,l} + (\prod \tilde{n} - 1)n_{h,l,Q}$  times where  $C$  calls the  $l$ -th hash oracle at most  $n_{h,l}$  times, and its other parameters are the same as those of  $C$ . Therefore, the context  $f(C)[C''_{\tilde{a}}[C'_{\tilde{a}}]]$  has the same parameters as the context  $C'$  in the statement of the lemma, so  $p'(C) = \prod \tilde{n} \times p(C')$ .  $\square$

*Proof of Lemma 6, Property 2.* Let  $Q_{\tilde{a}} = \text{AddIdx}(\tilde{a}, Q)$ . Let  $V_{\tilde{a}}$  be obtained by adding indices  $\tilde{a}$  to all variable names in  $V$ , and  $V' = \bigcup_{\tilde{a} \leq \tilde{n}} V_{\tilde{a}}$ . Let  $\text{corr}_{\tilde{a}} = \text{AddIdx}(\tilde{a}, \text{corr})$ . Let  $\text{corr}' = \text{AddSid}(T_{\text{sid}}, \text{corr})$ . Let  $C$  be an evaluation context acceptable for  $C_h[Q!]$  with public variables  $V$  that does not contain events used by  $\text{corr}'$ . Let  $C_1$  be obtained from  $C$  by renaming the variables  $y[\tilde{a}, \tilde{b}]$  to variables  $y_{\tilde{a}}[\tilde{b}]$  for  $y \in V$  and  $\tilde{a} \leq \tilde{n}$ . Let

$$\begin{aligned} C_2 = C_1 & [\text{newChannel } c; (\tilde{l} \leq \tilde{n} \text{ c}'[\tilde{i}](x : T_{\text{sid}}); \\ & \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(x[\tilde{i}'], x'[\tilde{i}']) \wedge x = x[\tilde{i}'] \\ & \text{then yield} \\ & \text{else let } x' = \text{cst in } \overline{c[\tilde{i}]} \langle \rangle \mid [])] \end{aligned}$$

Let  $C_{\tilde{a}} = (\prod_{\tilde{a}' < \tilde{n}, \tilde{a}' \neq \tilde{a}} Q_{\tilde{a}'}) \mid []$ . We have

$$\begin{aligned} \text{Adv}_{C_h[Q!]}^{\text{corr}'}(C) &= \Pr[C[C_h[Q!]] : \neg \text{corr}'] \\ &= \Pr[C_2[C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]] : \neg \forall \tilde{a} \leq \tilde{n}, \text{corr}_{\tilde{a}}] \end{aligned}$$

This equality of probabilities is shown by establishing a correspondence between the traces of  $C[C_h[Q!]]$  and the traces of  $C_2[C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]]$ : we eliminate communications on  $c$  and map variables  $y[\tilde{a}, \tilde{b}]$  to  $y_{\tilde{a}}[\tilde{b}]$  for  $y \in \text{var}(Q)$ , table entries  $\text{Tbl}(a', \tilde{b})$  to  $\text{Tbl}_{\tilde{a}}(\tilde{b})$  for tables  $\text{Tbl}$  of  $Q$ , and events  $e(a', \tilde{b})$  to events  $e_{\tilde{a}}(\tilde{b})$  for events  $e$  that occur in  $\text{corr}'$  (all other events can be ignored), where  $\tilde{a}$  is the index such that  $x[\tilde{a}] = a'$  and  $x'[\tilde{a}]$  is defined. Therefore, we have

$$\begin{aligned} \text{Adv}_{C_h[Q!]}^{\text{corr}'}(C) &\leq \sum_{\tilde{a} \leq \tilde{n}} \Pr[C_2[C_h[\prod_{\tilde{a} \leq \tilde{n}} Q_{\tilde{a}}]] : \neg \text{corr}_{\tilde{a}}] \\ &\leq \sum_{\tilde{a} \leq \tilde{n}} \Pr[C_2[C_h[C'_{\tilde{a}}[Q_{\tilde{a}}]]] : \neg \text{corr}_{\tilde{a}}] \end{aligned}$$

By Lemma 4 applied with context  $C_{\tilde{a}}$ , there exists an evaluation context  $C'_{\tilde{a}}$  such that  $C_h[C_{\tilde{a}}[Q_{\tilde{a}}]] \approx_0^{V'} C'_{\tilde{a}}[C_h[Q_{\tilde{a}}]]$ , so

$$\begin{aligned} \text{Adv}_{C_h[Q!]}^{\text{corr}'}(C) &\leq \sum_{\tilde{a} \leq \tilde{n}} \Pr[C_2[C'_{\tilde{a}}[C_h[Q_{\tilde{a}}]]] : \neg \text{corr}_{\tilde{a}}] \\ &\leq \sum_{\tilde{a} \leq \tilde{n}} p(C_2[C'_{\tilde{a}}]) \end{aligned}$$

since  $C_2[C'_{\tilde{a}}]$  does not contain events with indices  $\tilde{a}$ , so does not contain events used by  $\text{corr}_{\tilde{a}}$ . Moreover, the context  $C_2[C'_{\tilde{a}}]$  runs in time at most  $t_C + (\prod \tilde{n} - 1)t_Q$ , calls the  $l$ -th hash oracle at most  $n'_{h,l} = n_{h,l} + (\prod \tilde{n} - 1)n_{h,l,Q}$  times, and its other parameters are the same as those of  $C$ . Therefore, the context  $C_2[C'_{\tilde{a}}]$  has the same parameters as the context  $C'$  in the statement of the lemma, so  $\text{Adv}_{C_h[Q!]}^{\text{corr}'}(C) \leq \prod \tilde{n} \times p(C') = p'(C)$ . Hence,  $C_h[Q!]$  satisfies  $\text{corr}' = \text{AddSid}(T_{\text{sid}}, \text{corr})$  with public variables  $V$  up to probability  $p'$ .  $\square$

## A.9 Proof for Section 5.4

*Proof of Theorem 2.* Let us first introduce general notations. Given a process  $Q$  that makes all its inputs and outputs on distinct channels with indices the current replication indices, let  $\text{ch}_{\text{in}}(Q)$  (resp.  $\text{ch}_{\text{out}}(Q)$ ) be the channels on which  $Q$  performs its inputs (resp. outputs), and  $\text{ch}(Q) = \text{ch}_{\text{in}}(Q) \cup \text{ch}_{\text{out}}(Q)$ . For each channel  $c$  in  $\text{ch}(Q)$ , let  $\text{fr}_c$  be a fresh channel and  $x_c$  and  $x'_c$  be distinct fresh variables. Let  $\text{ch}'(Q) = \{\text{fr}_c \mid c \in \text{ch}(Q)\}$ . In  $Q$ , an input or output on channel  $c$  is under  $!^{i_c \leq \tilde{n}_c}$ , and has type  $\mathsf{T}_c$ . We define two relay processes as follows:

$$\begin{aligned} \text{Relay}(Q, \widetilde{M}, \widetilde{M}') &= \prod_{c \in \text{ch}_{\text{in}}(Q)} !^{i_c \leq \tilde{n}_c} \text{fr}_c[\widetilde{M}, \widetilde{i}_c](x_c : \mathsf{T}_c); \overline{c[\widetilde{M}', \widetilde{i}_c]} \langle x_c \rangle \mid \\ &\quad \prod_{c \in \text{ch}_{\text{out}}(Q)} !^{i_c \leq \tilde{n}_c} c[\widetilde{M}', \widetilde{i}_c](x_c : \mathsf{T}_c); \overline{\text{fr}_c[\widetilde{M}, \widetilde{i}_c]} \langle x_c \rangle \\ \text{Relay}'(Q, \widetilde{M}) &= \prod_{c \in \text{ch}_{\text{in}}(Q)} !^{i_c \leq \tilde{n}_c} c[\widetilde{M}, \widetilde{i}_c](x'_c : \mathsf{T}_c); \overline{\text{fr}_c[\widetilde{M}, \widetilde{i}_c]} \langle x'_c \rangle \mid \\ &\quad \prod_{c \in \text{ch}_{\text{out}}(Q)} !^{\widetilde{M} \leq \tilde{n}'} !^{i_c \leq \tilde{n}_c} \text{fr}_c[\widetilde{M}, \widetilde{i}_c](x'_c : \mathsf{T}_c); \overline{c[\widetilde{M}, \widetilde{i}_c]} \langle x'_c \rangle \end{aligned}$$

The goal of these relay processes is to renumber the first channel indices in  $Q$  from  $\widetilde{M}'$  to  $\widetilde{M}$ . However, to avoid confusions between channels before renumbering and channels after renumbering, we introduce fresh channels. So the relay process  $\text{Relay}(Q, \widetilde{M}, \widetilde{M}')$  performs the renumbering and forwards messages on channels  $\text{fr}_c$  for  $c \in \text{ch}_{\text{in}}(Q)$  to  $c$  and forward the replies on channels  $c \in \text{ch}_{\text{out}}(Q)$  back on  $\text{fr}_c$ . The relay process  $\text{Relay}'(Q, \widetilde{M})$  just performs the inverse renaming of channels: it forwards messages on channels  $c \in \text{ch}_{\text{in}}(Q)$  to  $\text{fr}_c$  and forwards the replies on channels  $\text{fr}_c$  for  $c \in \text{ch}_{\text{out}}(Q)$  back on  $c$ , so that after applying both relay processes, the messages are exchanged on channels in  $\text{ch}(Q)$  as in  $Q$ . The process  $\text{Relay}'(Q, \widetilde{M})$  does not renumber channel indices. We use the same notations for a context  $C$  instead of a process  $Q$ .

We can now start the proof itself. Let  $\tilde{u}$ ,  $\tilde{u}'$ , and  $\tilde{u}''$  be fresh variables. In particular, they are not in  $V_1$ . Let

$$\begin{aligned} C_5 &= \mathbf{newChannel} \text{ch}'(Q_{2B}); (!^{\tilde{i}' \leq \tilde{n}'} \text{Relay}'(Q_{2B}, \tilde{i}')) \mid \\ &\quad \mathbf{newChannel} \ c'_1, c_2, \text{ch}(Q_{2B}); \\ &\quad (C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \\ &\quad \quad \mathbf{find} \ \tilde{u}'' = \tilde{i}''' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}_A[\tilde{i}'''], k'_A[\tilde{i}''']) \wedge \\ &\quad \quad \text{sid}(\widetilde{msg}_A) = \text{sid}(\widetilde{msg}_A[\tilde{i}''']) \ \mathbf{then} \ \mathbf{yield} \ \mathbf{else} \\ &\quad \quad \mathbf{let} \ k'_A = k_A \ \mathbf{in} \ \overline{c'_1[\tilde{i}]} \langle \text{sid}(\widetilde{msg}_A) \rangle; c_2[\tilde{i}](); \overline{c'_A[\tilde{i}]} \langle M_A \rangle; Q_{1A}, \\ &\quad \mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_B), k_B); \\ &\quad \quad \mathbf{find} \ \tilde{u} = \tilde{i}'' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}_A[\tilde{i}''], k'_A[\tilde{i}'']) \wedge \\ &\quad \quad \text{sid}(\widetilde{msg}_A[\tilde{i}'']) = \text{sid}(\widetilde{msg}_B) \wedge \mathbf{fresh}(\tilde{i}'', \tilde{u}) \\ &\quad \quad \mathbf{then} \ \overline{c'_B[\tilde{i}']} \langle M_B \rangle; (Q_{1B} \mid \text{Relay}(Q_{2B}, \tilde{i}', \tilde{u}))]) \\ &\quad \mid [])) \end{aligned}$$

where  $\mathbf{fresh}(\tilde{i}'', \tilde{u}) = \mathbf{find} \ \tilde{u}' = \tilde{i}''' \leq \tilde{n}' \ \mathbf{suchthat} \ \mathbf{defined}(\tilde{u}[\tilde{i}''']) \wedge \tilde{u}[\tilde{i}'''] = \tilde{i}'' \ \mathbf{then} \ \mathbf{false} \ \mathbf{else} \ \mathbf{true}$ . We have  $\mathbf{fresh}(\tilde{i}'', \tilde{u})$  when  $\tilde{i}''$  was not used before, that is, it does not occur in the array  $\tilde{u}$ .

Let

$$\begin{aligned}
G_0 = C_h'''[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \\
& \text{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \text{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k'_A[\widetilde{i}''']) \wedge \\
& \quad \text{sid}(\widetilde{msg}_A) = \text{sid}(\widetilde{msg}_A[\widetilde{i}''']) \text{ then yield else} \\
& \text{let } k'_A = k_A \text{ in let } x = \text{sid}(\widetilde{msg}_A) \text{ in} \\
& \text{new } k : T; \overline{c'_A[\widetilde{i}]} \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}), \\
& \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \\
& \text{find } \widetilde{u} = \widetilde{i}'' \leq \widetilde{n} \text{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}''], k'_A[\widetilde{i}''], k[\widetilde{i}''], x[\widetilde{i}'']) \wedge \\
& \quad \text{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \text{sid}(\widetilde{msg}_B) \wedge \text{fresh}(\widetilde{i}'', \widetilde{u}) \text{ then} \\
& \quad \overline{c'_B[\widetilde{i}']} \langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, x[\widetilde{u}]/x\})]]
\end{aligned}$$

and

$$\begin{aligned}
G_1 = C_h'''[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \\
& \text{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \text{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k[\widetilde{i}''']) \wedge \\
& \quad \text{sid}(\widetilde{msg}_A) = \text{sid}(\widetilde{msg}_A[\widetilde{i}''']) \text{ then yield else} \\
& \text{new } k : T; \overline{c'_A[\widetilde{i}]} \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{\text{sid}(\widetilde{msg}_A)/x\}), \\
& \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \\
& \text{find } \widetilde{u} = \widetilde{i}'' \leq \widetilde{n} \text{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}''], k[\widetilde{i}'']) \wedge \\
& \quad \text{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \text{sid}(\widetilde{msg}_B) \wedge \text{fresh}(\widetilde{i}'', \widetilde{u}) \text{ then} \\
& \quad \overline{c'_B[\widetilde{i}']} \langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, \text{sid}(\widetilde{msg}_B)/x\})]]
\end{aligned}$$

The games  $G_0$  and  $G_1$  run the key exchange protocol followed by the protocol that uses the key, much like  $S_{composed}$ . However, in the participant  $A$  (after event  $e_A$ ), they do not run the protocol that uses the key when the same session identifier has already been seen in a previous session (**find**  $\widetilde{u}''$ ), and they generate a fresh key  $k$  instead of using the key provided by the key exchange protocol (**new**  $k : T$ ). In the participant  $B$  (after event  $e_B$ ), they get the key that has been generated in  $A$  with the same session identifier (**find**  $\widetilde{u}$ ), and require that the key of a given session of  $A$  is reused at most once by  $B$  (condition  $\text{fresh}(\widetilde{i}'', \widetilde{u})$ ).

Let  $Q'_2 = \text{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\text{sid}}, Q_2)$ , so that  $S_2 = C_h'[Q'_2]$ . We prove that  $G_1 \xrightarrow{V_1, V_1 \cup \{\widetilde{u}\}}_{f', 0} C_h'''[C_5[Q'_2]]$ , using  $G_0$  as intermediate game.

In the game  $C_h'''[C_5[Q'_2]]$ , the **find**  $\widetilde{u}''$  in  $C_5$  guarantees that the same value of  $x = \text{sid}(\widetilde{msg}_A)$  is never seen twice on channel  $c'_1$ , so the **find** introduced at the root of  $Q'_2$  by  $\text{AddReplSid}$  never succeeds. Then we can remove this **find**, keeping only its **else** branch. We can also remove the assignment **let**  $x' = \text{cst}$  since  $x'$  is now unused. Let  $Q''_{2B} = \text{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\text{sid}}, Q_{2B})$  and  $Q''_2 = \widetilde{i} \leq \widetilde{n} \cdot c'_1[\widetilde{i}](x : T_{\text{sid}}); \text{new } k : T; \overline{c_2[\widetilde{i}]} \langle \rangle; (Q'_{2A} \mid Q''_{2B})$ . By the previous reasoning, we have  $C_h'''[C_5[Q'_2]] \approx_0^{V_1 \cup \{\widetilde{u}\}} C_h'''[C_5[Q''_2]]$ .

Second, we show  $G_0 \xrightarrow{V_1, V_1 \cup \{\widetilde{u}\}}_{f', 0} C_h'''[C_5[Q''_2]]$ . To prove this property, let us consider an evaluation context  $C_6$  acceptable for  $G_0$  with public variables  $V_1$ . Let  $f'(C_6) = C'_6$  be obtained from  $C_6$  by replacing array accesses  $y[\widetilde{M}, \widetilde{M}']$  with  $y[\widetilde{u}[\widetilde{M}], \widetilde{M}']$ , when  $y \in V_1$  is defined in  $Q'_{2B}$ ,  $\widetilde{M}$  contains as many elements as  $\widetilde{i}'$ , and  $\widetilde{M}'$  contains the other indices of  $y$  if any. The context  $C'_6$  is

an evaluation context acceptable for  $C_h''[C_5[Q_2'']]$  with public variables  $V_1 \cup \{\tilde{u}\}$ . We establish a correspondence between the traces of  $C_6[G_0]$  and those of  $C_6'[C_h''[C_5[Q_2'']]]$ : we eliminate communications on the private channels  $c_1'$ ,  $c_2$ ,  $\text{ch}(Q_{2B})$ , and  $\text{ch}'(Q_{2B})$  (Appendix A.2) and we renumber the variables of  $Q_{2B}'$ , replacing indices  $\tilde{a} \leq \tilde{n}'$  in  $C_6[G_0]$  with  $\tilde{u}[\tilde{a}] \leq \tilde{n}$  in  $C_6'[C_h''[C_5[Q_2'']]]$ . (The condition  $\text{fresh}(\tilde{i}'', \tilde{u})$  guarantees that  $\tilde{u}$  never takes twice the same value, hence the function from  $\tilde{a}$  to  $\tilde{u}[\tilde{a}]$  is injective. We exclude **defined** conditions in  $Q_2$  to facilitate this renumbering.) In this correspondence between traces, an execution of  $Q_{2B}' = \text{AddIdxSid}(\tilde{i}' \leq \tilde{n}', x : T_{\text{sid}}, Q_{2B})$  with replication indices  $\tilde{a}$  in  $C_6[G_0]$  corresponds to an execution of  $Q_{2B}'' = \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_{2B})$  with replication indices  $\tilde{u}[\tilde{a}]$  in  $C_6'[C_h''[C_5[Q_2'']]]$ . These two executions have the same values of  $k$  and  $x$  due to the substitution  $k[\tilde{u}]/k, x[\tilde{u}]/x$  in  $G_0$ . They have the same value of other variables by the renumbering of variables, and produce the same events and table entries, since the events and table entries do not contain the replication indices  $\tilde{i}$  but the session identifier  $x$ . The channel indices are renumbered using the relay processes  $\text{Relay}(Q_{2B}, \tilde{i}', \tilde{u})$  and  $\text{Relay}'(Q_{2B}, \tilde{i}')$ . In  $C_h''[C_5[Q_2'']]$ , in **find**  $\tilde{u}$ , when  $k'_A[\tilde{i}']$  is defined, the output on  $c_1'[\tilde{i}']$  has been executed, so  $x[\tilde{i}']$  and  $k[\tilde{i}']$  are defined, the input on  $c_2[\tilde{i}']$  has been executed, and the process  $Q_{2B}'$  is available with indices  $\tilde{u} = \tilde{i}''$ . Moreover,  $Q_{2B}''$  with these indices is not available earlier to the context, because the channels of  $Q_{2B}''$  are hidden by **newChannel** and are accessible only via the relay processes  $\text{Relay}(Q_{2B}, \tilde{i}', \tilde{u})$  and  $\text{Relay}'(Q_{2B}, \tilde{i}')$ . In  $G_0$ , the variables  $k'_A$ ,  $x$ , and  $k$  are always defined at the same time. Hence, in both  $C_h''[C_5[Q_2'']]$  and  $G_0$ , when  $k'_A[\tilde{i}']$  is defined, so are  $k[\tilde{i}']$  and  $x[\tilde{i}']$ . We add them to the **defined** condition so that we can refer to  $k[\tilde{u}]$  and  $x[\tilde{u}]$  in  $Q_{2B}'\{k[\tilde{u}]/k, x[\tilde{u}]/x\}$  in  $G_0$ .

Third, we show  $G_1 \approx_0^{V_1} G_0$ . Starting from  $G_0$ , we replace  $x$  with its value  $\text{sid}(\widetilde{\text{msg}}_A)$ , and we note that  $x[\tilde{u}] = \text{sid}(\widetilde{\text{msg}}_A[\tilde{u}]) = \text{sid}(\widetilde{\text{msg}}_B)$  by the condition of **find**  $\tilde{u}$ , so we replace  $x[\tilde{u}]$  with  $\text{sid}(\widetilde{\text{msg}}_B)$ . Since  $k'_A$  and  $x$  are defined at the same time as  $k$ , we replace  $k'_A$  and  $x$  with  $k$  in the **defined** conditions. Finally, we can remove the definitions of  $k'_A$  and  $x$  since they are now unused, and we obtain the game  $G_1$ .

By combining the previous three results, we have  $G_1 \xrightarrow[f',0]{V_1, V_1 \cup \{\tilde{u}\}} C_h''[C_5[Q_2']]$ . Let  $C_5^\circ$  be obtained from  $C_5$  by removing all events and  $G_1^\circ$  be obtained from  $G_1$  by removing all events of  $S_1$ . By Lemma 11, we have

$$G_1^\circ \xrightarrow[f',0]{V_1, V_1 \cup \{\tilde{u}\}} C_h''[C_5^\circ[Q_2']] \quad (10)$$

By Lemma 4, there exists an evaluation context  $C_5'$  such that

$$C_h''[C_5^\circ[Q_2']] \approx_0^{V_1 \cup \{\tilde{u}\}} C_5'[C_h'[Q_2']] = C_5'[S_2] \quad (11)$$

where the context  $C_5'$  runs in time at most  $t_{C_5^\circ} \leq t_1$ , calls the  $l$ -th hash oracle in  $C_h'$  at most  $n_{h,l,C_5^\circ} \leq n_{h,l,1}$  times, so  $n'_{h,l} = n''_{h,l} + n_{h,l,1}$ , and its other parameters are the same as those of  $C_5^\circ$ .

The proof that  $S_{\text{composed}}^\circ \xrightarrow[f,p_3]{V_1, V_2} S_2$  then proceeds in two main steps:

1. First, we write the process  $G_1$  above as an evaluation context around

$$S_1' = C_h[C[\text{event } e_A(\text{sid}(\widetilde{\text{msg}}_A), k_A, \tilde{i}); \text{new } k'_A : T; \overline{c_A''[\tilde{i}]} \langle (k'_A, M_A) \rangle); Q_{1A}, \\ \text{event } e_B(\text{sid}(\widetilde{\text{msg}}_B), k_B); \overline{c_B[\tilde{i}']} \langle M_B \rangle; Q_{1B}]]$$

Let  $S_1''$  be obtained by replacing **new**  $k'_A : T$  with **let**  $k'_A = k_A$  **in** in  $S_1'$ . Let  $G_2$  be obtained by replacing **new**  $k : T$  with **let**  $k = k_A$  **in** in  $G_1$ . Let  $G_2^\circ$ ,  $S_1'^\circ$ , and  $S_1''^\circ$  be obtained from  $G_2$ ,  $S_1'$ , and  $S_1''$  respectively by removing all events of  $S_1$ . Since  $S_1$  preserves the secrecy of  $k'_A$  with public variables  $V$  ( $k_A, k'_A \notin V$ ) up to probability  $p$ , by Lemma 9,  $S_1'^\circ$  is indistinguishable from  $S_1''^\circ$ , so  $G_1^\circ$  is indistinguishable from  $G_2^\circ$ .

2. Second, we write  $G_2$  as an evaluation context around  $S_1$ . Since  $S_1$  satisfies the correspondences (2) and (3), so does  $G_2$ , so we obtain that, up to a small probability, the **find**  $\tilde{u}''$  in  $G_2$  fails and the **find**  $\tilde{u}$  succeeds with  $k[\tilde{u}] = k_B$ . From that, we show that  $G_2$  is indistinguishable from  $S_{composed}$ , so by Lemma 11,  $G_2^\circ$  is indistinguishable from  $S_{composed}^\circ$ .

We conclude by combining these results with (10) and (11). Let us detail these two steps.

Let  $\widetilde{msg}'_A$  (resp.  $\widetilde{msg}'_B$ ) be the sequence of variables corresponding to  $\widetilde{msg}_A$  (resp.  $\widetilde{msg}_B$ ), but using variables  $x_c$  of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  instead of variables of  $C$ . (We use  $\emptyset$  for the empty sequence. In case a variable of  $\widetilde{msg}_A$  is output by the output  $c_A[\tilde{i}]\langle M_A \rangle$ ,  $M_A$  is equal to this variable, and  $\widetilde{msg}'_A$  uses  $x_A$  instead of this variable. In the context  $C_1$  below,  $x_A$  contains the value of  $M_A$ .) The goal of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  is to capture in variables  $\widetilde{msg}'_A$ , visible outside  $C$ , the content of  $\widetilde{msg}_A$ , which are variables internal to  $C$  but sent or received on public channels, and similarly for  $\widetilde{msg}_B$ . Since we use the relay process  $\text{Relay}(C, \emptyset, \emptyset)$ , the external inputs and outputs are now done on the new channels  $\text{fr}_c$  for  $c \in \text{ch}(C)$ , so we rename the channels  $c \in \text{ch}(C)$  to  $\text{fr}_c$  on the other side of the equivalence below.

Given a process  $Q$  that makes all its inputs and outputs on distinct channels with indices the current replication indices, let  $\text{ch}_{\text{st}}(Q)$  be the channels on which  $Q$  performs its first inputs. For each channel  $c$  in  $\text{ch}_{\text{st}}(Q)$ , let  $\text{fr}_c$  be a fresh channel and  $x_c$  be a fresh variable. In  $Q$ , an input on channel  $c$  is under  $!^{i_c \leq \tilde{n}_c}$ , and has type  $T_c$ . We define a relay process as follows:

$$\text{Relay}_{\text{st}}(Q) = \prod_{c \in \text{ch}_{\text{st}}(Q)} !^{i_c \leq \tilde{n}_c} \text{fr}_c[\tilde{i}, \tilde{i}_c](x_c : T_c); \overline{c}[\tilde{i}, \tilde{i}_c]\langle x_c \rangle$$

The process  $\text{Relay}_{\text{st}}(Q)$  relays messages received on fresh channels  $\text{fr}_c$  for  $c \in \text{ch}_{\text{st}}(Q)$  to channel  $c$ . In the context  $C_1$  below, we hide the channels in  $\text{ch}_{\text{st}}(Q_{1A})$  and execute  $\text{Relay}_{\text{st}}(Q_{1A})$  only when the **find**  $\tilde{u}''$  fails. As a result, the adversary against  $C_1[S'_1]$  can access  $Q_{1A}$  (by sending messages on  $\text{fr}_{ch}$  for  $c \in \text{ch}_{\text{st}}(Q_{1A})$ ) only when the **find**  $\tilde{u}''$  fails, similarly to what happens in  $G_1$ . We proceed in a similar way for  $Q_{1B}$ .

Suppose that  $M_A$  has type  $T_A$  and  $M_B$  has type  $T_B$ . Let  $c''_A$  be a fresh channel. Let  $k'$ ,  $x_A$ ,  $x_B$  be fresh variables. Let

$$\begin{aligned} C_1 = & \text{newChannel } c''_A, c_B, \text{ch}(C), \text{ch}_{\text{st}}(Q_{1A}), \text{ch}_{\text{st}}(Q_{1B}); \\ & (\text{Relay}(C, \emptyset, \emptyset) \mid \\ & !^{i \leq \tilde{n}} c''_A[\tilde{i}]((k' : T, x_A : T_A)); \\ & \text{find } \tilde{u}'' = \tilde{i}''' \leq \tilde{n} \text{ suchthat defined}(\widetilde{msg}'_A[\tilde{i}'''], k[\tilde{i}'''], \widetilde{msg}'_A[\tilde{i}]) \wedge \\ & \text{sid}(\widetilde{msg}'_A[\tilde{i}]) = \text{sid}(\widetilde{msg}'_A[\tilde{i}''']) \text{ then yield else} \\ & \text{let } k = k' \text{ in } \overline{c''_A}[\tilde{i}]\langle x_A \rangle; (\text{Relay}_{\text{st}}(Q_{1A}) \mid Q'_{2A}\{\text{sid}(\widetilde{msg}'_A[\tilde{i}])/x\}) \mid \\ & !^{i' \leq \tilde{n}'} c_B[\tilde{i}'](x_B : T_B); \\ & \text{find } \tilde{u} = \tilde{i}'' \leq \tilde{n} \text{ suchthat defined}(\widetilde{msg}'_A[\tilde{i}''], k[\tilde{i}''], \widetilde{msg}'_B[\tilde{i}']) \wedge \\ & \text{sid}(\widetilde{msg}'_A[\tilde{i}'']) = \text{sid}(\widetilde{msg}'_B[\tilde{i}']) \wedge \text{fresh}(\tilde{i}'', \tilde{u}) \text{ then} \\ & \overline{c'_B}[\tilde{i}']\langle x_B \rangle; (\text{Relay}_{\text{st}}(Q_{1B}) \mid Q'_{2B}\{k[\tilde{u}]/k, \text{sid}(\widetilde{msg}'_B[\tilde{i}'])/x\}) \mid \\ & []) \\ Q'_1 = & C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \text{new } k'_A : T; \overline{c''_A}[\tilde{i}]\langle (k'_A, M_A) \rangle; Q_{1A}, \\ & \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c_B}[\tilde{i}']\langle M_B \rangle; Q_{1B}] \end{aligned}$$

Let  $G_2$  be obtained by replacing **new**  $k : T$  with **let**  $k = k_A$  **in** in  $G_1$ :

$$\begin{aligned}
G_2 = & C_h''[C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \\
& \mathbf{find} \ \tilde{u}'' = \tilde{i}''' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}_A[\tilde{i}'''], k[\tilde{i}''']) \wedge \\
& \text{sid}(\widetilde{msg}_A) = \text{sid}(\widetilde{msg}_A[\tilde{i}''']) \ \mathbf{then} \ \mathbf{yield} \ \mathbf{else} \\
& \mathbf{let} \ k = k_A \ \mathbf{in} \ \overline{c'_A[\tilde{i}]} \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{\text{sid}(\widetilde{msg}_A)/x\}), \\
& \mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_B), k_B); \\
& \mathbf{find} \ \tilde{u} = \tilde{i}'' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}_A[\tilde{i}''], k[\tilde{i}'']) \wedge \\
& \text{sid}(\widetilde{msg}_A[\tilde{i}'']) = \text{sid}(\widetilde{msg}_B) \wedge \mathbf{fresh}(\tilde{i}'', \tilde{u}) \ \mathbf{then} \\
& \overline{c'_B[\tilde{i}']} \langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k[\tilde{u}]/k, \text{sid}(\widetilde{msg}_B)/x\})]]
\end{aligned}$$

Let  $Q'_1$  be obtained by replacing **new**  $k'_A : T$  with **let**  $k'_A = k_A$  **in** in  $Q'_1$ . Let  $G_2^\circ$ ,  $Q_1'^\circ$ , and  $Q_1''^\circ$  be obtained from  $G_2$ ,  $Q'_1$ , and  $Q_1''$  respectively by removing all events of  $S_1$ .

We have  $G_1\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_h''[C_1[Q'_1]]$ , by eliminating communications on  $c'_A$ ,  $c_B$ ,  $\text{ch}(C)$ ,  $\text{ch}_{\text{st}}(Q_{1A})$ , and  $\text{ch}_{\text{st}}(Q_{1B})$  (Appendix A.2), since the **finds** in  $C_1$  have the same effect as the ones in  $G_1$ , because  $\widetilde{msg}'_A$  and  $\widetilde{msg}'_B$  contain the same value as  $\widetilde{msg}_A$  and  $\widetilde{msg}_B$  respectively, by construction. Since  $C_1$  does not contain events of  $S_1$ , by Lemma 11,

$$G_1^\circ\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_h''[C_1[Q_1'^\circ]] \quad (12)$$

Similarly,  $G_2\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_h''[C_1[Q_1'']]$  so by Lemma 11,

$$G_2^\circ\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_h''[C_1[Q_1''^\circ]] \quad (13)$$

By Lemma 4, there exists an evaluation context  $C'_1$  such that

$$C_h''[C_1[Q_1'^\circ]] \approx_0^{V_1} C'_1[C_h[Q_1'^\circ]] \quad (14)$$

$$C_h''[C_1[Q_1''^\circ]] \approx_0^{V_1} C'_1[C_h[Q_1''^\circ]] \quad (15)$$

where the context  $C'_1$  runs in time at most  $t_{C_1} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_1} \leq n_{h,l,2}$  times, so  $n_{h,l} = n_{h,l}'' + n_{h,l,2}$ , and its other parameters are the same as those of  $C_1$ .

Since  $S_1 = C_h[Q_1]$  preserves the secrecy of  $k'_A$  with public variables  $V$  up to probability  $p$ , by Lemma 9, we have  $C_h[Q_1'^\circ] \approx_{p_0}^V C_h[Q_1''^\circ]$  where  $p_0(C_3, t_D) = p(C_3 + t_D)$ . Therefore,

$$C'_1[C_h[Q_1'^\circ]] \approx_{p_1}^{V_1} C'_1[C_h[Q_1''^\circ]] \quad (16)$$

where  $p_1(C_3, t_D) = p_0(C_3[C'_1], t_D) = p(C_3[C'_1] + t_D) = p(C'_3 + t_D)$  and the context  $C'_3$  runs in time at most  $t_{C_3} + t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l} = n_{h,l}'' + n_{h,l,2}$  times, and its other parameters are the same as those of  $C_3$ .

By combining (12), (14), (16), (15), and (13) by transitivity, we obtain

$$G_1^\circ\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\} \approx_{p_1}^{V_1} G_2^\circ\{\text{fr}_c/c, c \in \text{ch}(C) \cup \text{ch}_{\text{st}}(Q_{1A}) \cup \text{ch}_{\text{st}}(Q_{1B})\}$$

so by renaming channels

$$G_1^\circ \approx_{p_1}^{V_1} G_2^\circ \quad (17)$$

Introducing relay processes and renaming channels as above, we define

$$\begin{aligned}
C_2 = & \mathbf{newChannel} \ c_A, c_B, \mathbf{ch}(C), \mathbf{ch}_{\text{st}}(Q_{1A}), \mathbf{ch}_{\text{st}}(Q_{1B}); \\
& (\text{Relay}(C, \emptyset, \emptyset) \mid \\
& \quad !^{\tilde{i} \leq \tilde{n}} c_A[\tilde{i}](x_A : T_A); \\
& \quad \mathbf{find} \ \tilde{u}'' = \tilde{i}''' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}'_A[\tilde{i}'''], k[\tilde{i}'''], \widetilde{msg}'_A[\tilde{i}]) \wedge \\
& \quad \quad \mathbf{sid}(\widetilde{msg}'_A[\tilde{i}]) = \mathbf{sid}(\widetilde{msg}'_A[\tilde{i}''']) \ \mathbf{then} \ \mathbf{yield} \ \mathbf{else} \\
& \quad \mathbf{if} \ \mathbf{defined}(k'_A[\tilde{i}]) \ \mathbf{then} \ \mathbf{let} \ k = k'_A[\tilde{i}] \ \mathbf{in} \ \overline{c'_A[\tilde{i}]}(x_A); \\
& \quad (\text{Relay}_{\text{st}}(Q_{1A}) \mid Q'_{2A}\{\mathbf{sid}(\widetilde{msg}'_A[\tilde{i}])/x\}) \mid \\
& \quad !^{\tilde{i}' \leq \tilde{n}'} c_B[\tilde{i}'](x_B : T_B); \\
& \quad \mathbf{find} \ \tilde{u} = \tilde{i}'' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(\widetilde{msg}'_A[\tilde{i}''], k[\tilde{i}''], \widetilde{msg}'_B[\tilde{i}']) \wedge \\
& \quad \quad \mathbf{sid}(\widetilde{msg}'_A[\tilde{i}'']) = \mathbf{sid}(\widetilde{msg}'_B[\tilde{i}']) \wedge \mathbf{fresh}(\tilde{i}'', \tilde{u}) \ \mathbf{then} \\
& \quad \overline{c'_B[\tilde{i}']}(x_B); (\text{Relay}_{\text{st}}(Q_{1B}) \mid Q'_{2B}\{k[\tilde{u}]/k, \mathbf{sid}(\widetilde{msg}'_B[\tilde{i}'])/x\}) \mid \\
& \quad [])
\end{aligned}$$

so that we have

$$G_2\{\text{fr}_c/c, c \in \mathbf{ch}(C) \cup \mathbf{ch}_{\text{st}}(Q_{1A}) \cup \mathbf{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_h''[C_2[Q_1]]$$

By Lemma 4, there exists an evaluation context  $C_2'$  such that

$$C_h''[C_2[Q_1]] \approx_0^{V_1} C_2'[C_h[Q_1]] = C_2'[S_1]$$

where the context  $C_2'$  runs in time at most  $t_{C_2} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_2} \leq n_{h,l,2}$  times, so  $n_{h,l} = n_{h,l}'' + n_{h,l,2}$ , and its other parameters are the same as those of  $C_2$ . Therefore,

$$G_2\{\text{fr}_c/c, c \in \mathbf{ch}(C) \cup \mathbf{ch}_{\text{st}}(Q_{1A}) \cup \mathbf{ch}_{\text{st}}(Q_{1B})\} \approx_0^{V_1} C_2'[S_1]$$

The process  $S_1$  satisfies the correspondences (2) and (3) with public variables  $V \cup \{k'_A\}$  up to probabilities  $p'$  and  $p''$  respectively. Hence, by Lemma 1, the process  $C_2'[S_1]$  satisfies these correspondences with public variables  $V_1$  up to probabilities  $p'_1(C_3, t_D) = p'(C_3[C_2], t_D) = p'(C'_3, t_D)$  and  $p''_1(C_3, t_D) = p''(C_3[C'_2], t_D) = p''(C'_3, t_D)$  respectively, where the context  $C'_3$  runs in time at most  $t_{C_3} + t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l} = n_{h,l}'' + n_{h,l,2}$  times, and its other parameters are the same as those of  $C_3$ , and so does  $G_2$ .

In  $G_2$ , for traces that satisfy these two correspondences, we have that event  $e_A$  is never executed twice with the same session identifier  $\mathbf{sid}(\widetilde{msg}_A)$  because of the correspondence (3) so **find**  $\tilde{u}''$  always fails. It can then be removed, keeping only its **else** branch. Moreover, by (2), for each  $\tilde{i}'$  such that event  $e_B(\mathbf{sid}(\widetilde{msg}_B[\tilde{i}']), k_B[\tilde{i}'])$  has been executed, there exists a distinct  $\tilde{i}$  such that  $e_A(\mathbf{sid}(\widetilde{msg}_A[\tilde{i}]), k_A[\tilde{i}], \tilde{i})$  has been executed, with  $\mathbf{sid}(\widetilde{msg}_A[\tilde{i}]) = \mathbf{sid}(\widetilde{msg}_B[\tilde{i}'])$  and  $k_A[\tilde{i}] = k_B[\tilde{i}']$ . Therefore, the conditions of **find**  $\tilde{u}$  are satisfied with  $\tilde{i}'' = \tilde{i}$ , so **find**  $\tilde{u}$  succeeds. (Injectivity guarantees that we can always find a fresh  $\tilde{i}''$ .) Furthermore, by (3), there is at most one  $\tilde{i}$  such that  $e_A(\mathbf{sid}(\widetilde{msg}_A[\tilde{i}]), k_A[\tilde{i}], \tilde{i})$  is executed with  $\mathbf{sid}(\widetilde{msg}_A[\tilde{i}]) = \mathbf{sid}(\widetilde{msg}_B[\tilde{i}'])$ , so we have  $\tilde{i}'' = \tilde{i}$  and  $k[\tilde{u}] = k[\tilde{i}''] = k[\tilde{i}] = k_A[\tilde{i}] = k_B[\tilde{i}']$ . Therefore, we can run  $Q'_{2B}\{k_B[\tilde{i}']/k, \mathbf{sid}(\widetilde{msg}_B)/x\}$

when **find**  $\tilde{u}$  succeeds. Since  $\tilde{u}$  is no longer used, we can remove **find**  $\tilde{u}$ . Hence,

$$\begin{aligned} G_2 &\approx_{p'_1+p''_1}^{V_1} C_h''[C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \mathbf{let} \ k = k_A \ \mathbf{in} \ \overline{c'_A[\tilde{i}]} \langle M_A \rangle; \\ &\quad (Q_{1A} \mid Q'_{2A}\{\text{sid}(\widetilde{msg}_A)/x\}), \\ &\quad \mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\tilde{i}']} \langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \text{sid}(\widetilde{msg}_B)/x\})]] \\ &\approx_{p'_1+p''_1}^{V_1} S_{composed} \end{aligned}$$

By Lemma 11, we have

$$G_2^\circ \approx_{p'_1+p''_1}^{V_1} S_{composed}^\circ \quad (18)$$

so by combining (18), (17), (10), and (11), we obtain

$$S_{composed}^\circ \approx_{p'_1+p''_1}^{V_1} G_2^\circ \approx_{p_1}^{V_1} G_1^\circ \xrightarrow{f',0}^{V_1, V_1 \cup \{\tilde{u}\}} C_h''[C_5^\circ[Q'_2]] \approx_0^{V_1 \cup \{\tilde{u}\}} C'_5[S_2]$$

so  $S_{composed}^\circ \xrightarrow{f', p_1+p'_1+p''_1}^{V_1, V_1 \cup \{\tilde{u}\}} C'_5[S_2]$ , so we obtain  $S_{composed}^\circ \xrightarrow{f, p_3}^{V_1, V_2} S_2$  by defining  $f(C_3) = f'(C_3)[C'_5[[]]]$ .

Furthermore, by Lemma 10, if  $y \in V_2 \cap \text{var}(Q_{2A})$ , then  $f$  is secrecy-preserving for  $y \mapsto (y, f_{\text{sec}})$  where  $f_{\text{sec}}(C_3) = f'(C_3)[C'_5[[]]]$ . Let us verify the assumptions of Lemma 10. First, we have  $(\text{var}(f'(C_3)) \cup \text{var}(C'_5)) \cap \text{var}(R_y) = \emptyset$ . Indeed,  $\text{var}(C_3) \cap \text{var}(R_y) = \emptyset$  because  $C_3$  is an evaluation context acceptable for  $S_{composed}^\circ \mid R_y$  with public variables  $V_1 \setminus \{y\}$  and  $(V_1 \setminus \{y\}) \cap \text{var}(R_y) = \emptyset$  because the variables of  $R_y$  other than  $y$  are fresh. Moreover,  $\text{var}(C'_5) \cap \text{var}(R_y) \subseteq \text{var}(S_1) \cap \{y\} = \emptyset$  because  $y \in \text{var}(S_2)$  and  $S_1$  and  $S_2$  have no common variable, and the other variables are fresh: the variables of  $C'_5$  are those of  $C_5^\circ$  plus fresh variables by the construction done in Lemma 4, the variables of  $C_5^\circ$  are those of  $S_1$  plus the fresh variables  $\tilde{u}, \tilde{u}'$ , and the variables of  $R_y$  other than  $y$  are fresh. Second, let  $S_1^\circ$  be obtained from  $S_1$  by removing all events. We have  $\text{vardef}(f'(C_3)) \cap \text{var}(C'_5) = \text{vardef}(C_3) \cap \text{var}(S_1^\circ) \subseteq \text{vardef}(C_3) \cap \text{var}(S_{composed}^\circ \mid R_y) = \emptyset$  because, as above, the variables of  $C'_5$  are those of  $S_1^\circ$  plus fresh variables. Third,  $f'(C_3)$  and  $C'_5$  do not use any common table because  $C_3$  is an evaluation context acceptable for  $S_{composed}^\circ \mid R_y$  with public variables  $V_1 \setminus \{y\}$  so  $C_3$  and  $S_{composed}^\circ \mid R_y$  have no common table, the tables of  $f'(C_3)$  are those of  $C_3$ , and the tables of  $C'_5$  are those of  $C_5^\circ$  by the construction done in Lemma 4, which are among those of  $S_{composed}^\circ$ .

This property does not apply to variables  $y \in V_2 \cap \text{var}(Q_{2B})$  because they are renumbered by  $f$ , so we perform a separate proof for such variables. Let  $y \in V_2 \cap \text{var}(Q_{2B})$ . Suppose that  $S_2$  preserves the secrecy of  $y$  with public variables  $V_2 \setminus \{y\}$  up to probability  $p_2$ . By Lemma 1,  $C'_5[S_2]$  preserves the secrecy of  $y$  with public variables  $(V_2 \cup \text{var}(C'_5)) \setminus \{y\}$  up to probability  $p'_2$  such that  $p'_2(C_5) = p_1(C_5[C'_5])$ . Let  $R_y$  be the process used for testing secrecy of  $y$  in  $S_{composed}^\circ$ . Let  $R'_y$  be obtained by renumbering the variables  $R_y$ :  $R'_y = f'(R_y)$ , that is,

$$\begin{aligned} R'_y &= c_{s0}(); \mathbf{new} \ b : \text{bool}; \overline{c_{s0}} \langle \rangle; \\ &\quad (!^{i_s \leq n_s} c_s[i_s](\tilde{u}_1 \leq \tilde{n}', \tilde{u}_2 \leq \tilde{n}_2); \mathbf{if} \ \mathbf{defined}(y[\tilde{u}[\tilde{u}_1], \tilde{u}_2]) \ \mathbf{then} \\ &\quad \mathbf{if} \ b \ \mathbf{then} \ \overline{c_s[i_s]} \langle y[\tilde{u}[\tilde{u}_1], \tilde{u}_2] \rangle \ \mathbf{else} \\ &\quad \mathbf{find} \ u'_s = i'_s \leq n_s \ \mathbf{suchthat} \ \mathbf{defined}(y'[i'_s], \tilde{u}_1[i'_s], \tilde{u}_2[i'_s]) \wedge \tilde{u}_1[i'_s] = \tilde{u}_1 \wedge \tilde{u}_2[i'_s] = \tilde{u}_2 \\ &\quad \mathbf{then} \ \overline{c_s[i_s]} \langle y'[u'_s] \rangle \ \mathbf{else} \ \mathbf{new} \ y' : T; \overline{c_s[i_s]} \langle y' \rangle \\ &\quad \mid c'_s(b'); \mathbf{if} \ b = b' \ \mathbf{then} \ \mathbf{event\_abort} \ S \ \mathbf{else} \ \mathbf{event\_abort} \ \bar{S}) \end{aligned}$$



Let  $c_s''$  be a fresh channel,  $\tilde{u}_1', \tilde{u}_2', y'', \tilde{u}_1'', \tilde{u}_2'', y'''$  be fresh variables, and let us define relay processes:

$$\begin{aligned} \text{Relay}_R &= !^{i_s \leq n_s} c_s''[i_s](\tilde{u}_1' \leq \tilde{n}', \tilde{u}_2' \leq \tilde{n}_2); \text{if defined}(\tilde{u}[\tilde{u}_1']) \text{ then} \\ &\quad \overline{c_s[i_s]}(\tilde{u}[\tilde{u}_1'], \tilde{u}_2'); c_s[i_s](y'' : T); \overline{c_s''[i_s]}(y'') \\ \text{Relay}'_R &= !^{i_s \leq n_s} c_s[i_s](\tilde{u}_1'' \leq \tilde{n}', \tilde{u}_2'' \leq \tilde{n}_2); \overline{c_s''[i_s]}(\tilde{u}_1'', \tilde{u}_2''); c_s''[i_s](y''' : T); \overline{c_s[i_s]}(y''') \end{aligned}$$

The process  $\text{Relay}_R$  rennumbers the queries for  $y$  like  $f'$  rennumbers  $y$ . However, it uses channel  $c_s''$  instead of  $c_s$ , so we use process  $\text{Relay}'_R$  to revert to channel  $c_s$ . Let  $C_R = \mathbf{newChannel} \ c_s''; (\text{Relay}'_R \mid \mathbf{newChannel} \ c_s; (\text{Relay}_R \mid []))$ . Let  $R_y''$  be the process used for testing secrecy of  $y$  in  $C_5'[S_2]$ . (The process  $R_y''$  differs from  $R_y$  because  $y$  has indices  $\tilde{i}' \leq \tilde{n}', \tilde{i}_2 \leq \tilde{n}_2$  in  $S_{composed}^\circ$ , while it has indices  $\tilde{i} \leq \tilde{n}, \tilde{i}_2 \leq \tilde{n}_2$  in  $C_5'[S_2]$ .) We have

$$C_5'[S_2] \mid R_y' \approx_0^{V_1} C_R[C_5'[S_2] \mid R_y'']$$

by eliminating communications on the private channels  $c_s$  and  $c_s''$  (Appendix A.2). The equality test  $\tilde{u}_1[i_s'] = \tilde{u}_1$  performed by  $R_y'$  in the left-hand side becomes  $\tilde{u}[\tilde{u}_1[i_s']] = \tilde{u}[\tilde{u}_1]$  in the right-hand side because  $R_y''$  always receives  $\tilde{u}[\tilde{u}_1]$  instead of  $\tilde{u}_1$ . These two tests are equivalent, because  $\tilde{u}[\tilde{i}']$  cannot have the same value for different values of  $\tilde{i}'$  due to the condition  $\text{fresh}(\tilde{i}', \tilde{u})$ .

Let  $C_0$  be any evaluation context acceptable for  $S_{composed}^\circ \mid R_y$  with public variables  $V_1 \setminus \{y\}$ . The context  $f'(C_0)[C_R]$  is acceptable for  $C_5'[S_2] \mid R_y''$  with public variables  $(V_2 \cup \text{var}(C_5')) \setminus \{y\}$ , by the definition of acceptable evaluation contexts [14, Definition 4]. Indeed,

$$\begin{aligned} \text{var}(f'(C_0)[C_R]) \cap \text{var}(C_5'[S_2] \mid R_y'') &= \text{var}(C_0) \cap (\text{var}(C_5') \cup \text{var}(C_h') \cup \text{var}(Q_2) \cup \{y\}) \quad \text{because the other variables are fresh} \\ &\subseteq \text{var}(C_0) \cap (\text{var}(C_5') \cup ((\text{var}(C_h') \cup \text{var}(Q_2) \cup \{y\}) \cap V_1 \setminus \{y\})) \end{aligned} \tag{19}$$

$$\begin{aligned} &\subseteq \text{var}(C_0) \cap (\text{var}(C_5') \cup (V_2 \setminus \{y\})) \\ &\subseteq (\text{var}(C_5') \cup V_2) \setminus \{y\} \end{aligned} \tag{20}$$

The inclusion (19) holds because  $\text{var}(C_0) \cap (\text{var}(C_h') \cup \text{var}(Q_2) \cup \{y\}) \subseteq \text{var}(C_0) \cap \text{var}(S_{composed}^\circ \mid R_y) \subseteq V_1 \setminus \{y\}$  and (20) holds because  $y \notin \text{var}(C_0)$  by the inclusion  $\text{var}(C_0) \cap (\text{var}(Q_2) \cup \{y\}) \subseteq V_1 \setminus \{y\}$  shown above. We also have  $\text{vardef}(f'(C_0)[C_R]) \cap ((V_2 \cup \text{var}(C_5')) \setminus \{y\}) \subseteq \text{vardef}(C_0) \cap ((V_1 \cup \text{var}(S_{composed}^\circ \mid R_y)) \setminus \{y\}) = \emptyset$  because the variables of  $f'(C_0)[C_R]$  are those of  $C_0$  plus fresh variables,  $V_2 \subseteq V_1$ , the variables of  $C_5'$  are  $\text{var}(S_1^\circ) \subseteq \text{var}(S_{composed}^\circ)$  plus fresh variables,  $\text{vardef}(C_0) \cap (V_1 \setminus \{y\}) = \emptyset$ , and  $\text{vardef}(C_0) \cap \text{var}(S_{composed}^\circ \mid R_y) = \emptyset$ . Moreover,  $f'(C_0)[C_R]$  and  $C_5'[S_2] \mid R_y''$  do not use any common table, because  $C_0$  and  $S_{composed}^\circ \mid R_y$  do not use any common table, the tables of  $f'(C_0)[C_R]$  are those of  $C_0$ , and the tables of  $C_5'[S_2] \mid R_y''$  are those of  $C_5'$  and  $S_2$ , that is, of  $S_1$  and  $S_2$ , so of  $S_{composed}^\circ$ . We have

$$\begin{aligned} &\Pr[C_0[S_{composed}^\circ \mid R_y] : S] - \Pr[C_0[S_{composed}^\circ \mid R_y] : \bar{S}] \\ &\leq |\Pr[C_0[S_{composed}^\circ \mid R_y] : S] - \Pr[f'(C_0)[C_5'[S_2] \mid R_y'] : S]| + \\ &\quad (\Pr[f'(C_0)[C_5'[S_2] \mid R_y'] : S] - \Pr[f'(C_0)[C_5'[S_2] \mid R_y'] : \bar{S}]) + \\ &\quad |\Pr[f'(C_0)[C_5'[S_2] \mid R_y'] : \bar{S}] - \Pr[C_0[S_{composed}^\circ \mid R_y] : \bar{S}]| \\ &\leq p_3(C_0[[] \mid R_y], t_S) + \\ &\quad (\Pr[f'(C_0)[C_R[C_5'[S_2] \mid R_y'']] : S] - \Pr[f'(C_0)[C_R[C_5'[S_2] \mid R_y'']] : \bar{S}]) + \\ &\quad p_3(C_0[[] \mid R_y], t_{\bar{S}}) \end{aligned}$$

$$\begin{aligned} &\leq 2p_3(C_0[[\ ] \mid R_y], t_S) + p'_2(f'(C_0)[C_R]) \\ &\leq 2p_3(C_0[[\ ] \mid R_y], t_S) + p_2(f_{\text{sec}}(C_0)) \end{aligned}$$

since  $t_S = t_{\bar{S}}$  and  $p'_2(f'(C_0)[C_R]) = p_2(f'(C_0)[C_R[C'_5]]) = p_2(f_{\text{sec}}(C_0))$  by defining  $f_{\text{sec}}(C_0) = f'(C_0)[C_R[C'_5]]$ . We conclude that  $S_{\text{composed}}$  preserves the secrecy of  $y$  with public variables  $V_1 \setminus \{y\}$  up to probability  $p''$ , where  $p''(C_0) = 2p_3(C_0[[\ ] \mid R_y], t_S) + p_2(f_{\text{sec}}(C_0))$ . Therefore,  $f$  is secrecy-preserving for  $y \mapsto (y, f_{\text{sec}})$ .

Let us prove the second point. Let  $x_A, x_B, k''_A, k''_B$  be fresh variables not in  $V'$ . Let  $\widetilde{msg}'_A$  (resp.  $\widetilde{msg}'_B$ ) be the sequence of variables corresponding to  $\widetilde{msg}_A$  (resp.  $\widetilde{msg}_B$ ), but using variables  $x_c$  of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  instead of variables of  $C$ . As above, the goal of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  is to capture in variables  $\widetilde{msg}'_A$ , visible outside  $C$ , the content of  $\widetilde{msg}_A$ , which are variables internal to  $C$  but sent or received on public channels, and similarly for  $\widetilde{msg}_B$ . Since  $\text{Relay}(C, \emptyset, \emptyset)$  renames channels  $c \in \text{ch}(C)$  to  $fr_c$ , we use the relay process  $\text{Relay}'(C, \emptyset)$  to perform the reverse renaming.

Let

$$\begin{aligned} C_4 = & \text{newChannel } ch'(C); (\text{Relay}'(C, \emptyset) \mid \\ & \text{newChannel } c_A, c_B, ch(C); (\text{Relay}(C, \emptyset, \emptyset) \mid \\ & !\tilde{i} \leq \tilde{n}_{c_A}[\tilde{i}](x_A : T_A); \text{if defined}(k'_A[\tilde{i}], \widetilde{msg}'_A[\tilde{i}]) \text{ then} \\ & \quad \text{let } k''_A = k'_A[\tilde{i}] \text{ in } \overline{c'_A}[\tilde{i}]\langle x_A \rangle; Q'_{2A}\{k''_A/k, \text{sid}(\widetilde{msg}'_A[\tilde{i}])/x\} \mid \\ & !\tilde{i}' \leq \tilde{n}'_{c_B}[\tilde{i}'](x_B : T_B); \text{if defined}(k_B[\tilde{i}'], \widetilde{msg}'_B[\tilde{i}']) \text{ then} \\ & \quad \text{let } k''_B = k_B[\tilde{i}'] \text{ in } \overline{c'_B}[\tilde{i}']\langle x_B \rangle; Q'_{2B}\{k''_B/k, \text{sid}(\widetilde{msg}'_B[\tilde{i}'])/x\} \mid [])]) \end{aligned}$$

By Lemma 4, there exists an evaluation context  $C'_4$  such that

$$C''_h[C_4[Q_1]] \approx_0^{V'} C'_4[C_h[Q_1]] = C'_4[S_1]$$

where the context  $C'_4$  runs in time at most  $t_{C_4} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_4} \leq n_{h,l,2}$  times, so  $n_{h,l} = n''_{h,l} + n_{h,l,2}$ , and its other parameters are the same as those of  $C_4$ , that is, it does not alter the other parameters.

We have

$$\begin{aligned} C''_h[C_4[Q_1]] &\approx_0^{V'} C''_h[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \text{let } k'_A = k_A \text{ in if defined}(k'_A[\tilde{i}]) \text{ then} \\ & \quad \text{let } k''_A = k'_A[\tilde{i}] \text{ in } \overline{c'_A}[\tilde{i}]\langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k''_A/k, \text{sid}(\widetilde{msg}_A)/x\}), \\ & \quad \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \text{if defined}(k_B[\tilde{i}']) \text{ then} \\ & \quad \text{let } k''_B = k_B[\tilde{i}'] \text{ in } \overline{c'_B}[\tilde{i}']\langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k''_B/k, \text{sid}(\widetilde{msg}_B)/x\})]] \end{aligned}$$

by eliminating communications on the private channels  $c_A, c_B, ch(C), ch'(C)$  (Appendix A.2), so

$$\begin{aligned} C'_4[S_1] &\approx_0^{V'} C''_h[C_4[Q_1]] \\ &\approx_0^{V'} C''_h[C[\text{event } e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \overline{c'_A}[\tilde{i}]\langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \text{sid}(\widetilde{msg}_A)/x\}), \\ & \quad \text{event } e_B(\text{sid}(\widetilde{msg}_B), k_B); \overline{c'_B}[\tilde{i}']\langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \text{sid}(\widetilde{msg}_B)/x\})]] \\ &= S_{\text{composed}} \end{aligned}$$

since  $k'_A$  is an abbreviation for  $k'_A[\tilde{i}]$ , so  $k'_A[\tilde{i}]$  is always defined and  $k''_A = k'_A = k_A$ , and similarly for  $k_B$ . We can remove the assignment to  $k'_A$  since  $k'_A$  is not used and  $k'_A \notin V'$ .  $\square$

## A.10 Variant with Several Holes

In Theorem 2, event  $e_B$  appears in a single hole of the context  $C$ . Theorem 4 generalizes it to several holes.

**Theorem 4** (Variant with several holes for event  $e_B$ ). *Let  $C$  be any context with  $J + 1$  holes, with replications  $\tilde{i} \leq \tilde{n}$  above the first hole and  $\tilde{i}'_j \leq \tilde{n}'_j$  above the  $(j + 1)$ -th hole ( $1 \leq j \leq J$ ) and without **event\_abort**. Let  $Q_{1A}$  and  $Q_{1B,j}$  ( $1 \leq j \leq J$ ) be processes without **event\_abort**. Let  $k, k_A, k_{B,j}$  ( $1 \leq j \leq J$ ) be variables of type  $T$ . Let*

$$\begin{aligned} Q_1 &= C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \mathbf{let} \ k'_A = k_A \ \mathbf{in} \ \overline{c_A[\tilde{i}]} \langle M_A \rangle; Q_{1A}, \\ &\quad (\mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_{B,j}), k_{B,j}); \overline{c_{B,j}[\tilde{i}'_j]} \langle M_{B,j} \rangle; Q_{1B,j})_{1 \leq j \leq J}] \\ Q_2 &= c_1(); \mathbf{new} \ k : T; \overline{c_2} \langle \rangle; (Q_{2A} \mid Q_{2B}) \\ S_1 &= C_h[Q_1] \\ S_2 &= C'_h[\text{AddReplSid}(\tilde{i} \leq \tilde{n}, c'_1, T_{\text{sid}}, Q_2)] \end{aligned}$$

where  $Q_1$  and  $Q_2$  are hash-well-formed;  $\widetilde{msg}_A$  is a sequence of variables defined in  $C$  above the first hole and input or output by  $C$  above the first hole or by the output  $\overline{c_A[\tilde{i}]} \langle M_A \rangle$ ; for all  $j \in \{1, \dots, J\}$ ,  $\widetilde{msg}_{B,j}$  is a sequence of variables input or output by  $C$  above the  $(j + 1)$ -th hole; **sid** is a function that takes a sequence of messages and returns a session identifier of type  $T_{\text{sid}}$ ;  $C$ ,  $Q_{1A}$ ,  $Q_{1B,j}$  ( $1 \leq j \leq J$ ),  $Q_{2A}$ , and  $Q_{2B}$  make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;  $c_A$ ,  $c_{B,j}$  ( $1 \leq j \leq J$ ),  $c_1$ ,  $c'_1$ ,  $c_2$ ,  $k'_A$ ,  $e_A$ ,  $e_B$  do not occur elsewhere in  $S_1, S_2$ ;  $S_1$  and  $S_2$  have no common variable, no common channel, no common event, and no common table;  $S_1$  and  $S_2$  do not contain **newChannel**; and there is no **defined** condition in  $Q_2$ .

Let  $Q'_{2A} = \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_{2A})$  and  $Q'_{2B,j} = \text{AddIdxSid}(\tilde{i}'_j \leq \tilde{n}'_j, x : T_{\text{sid}}, Q_{2B})$ . Let  $c'_A, c'_{B,j}$  ( $1 \leq j \leq J$ ) be fresh channels. Let

$$\begin{aligned} Q_{\text{composed}} &= \\ C[\mathbf{event} \ e_A(\text{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \overline{c'_A[\tilde{i}]} \langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \text{sid}(\widetilde{msg}_A)/x\}), \\ &\quad (\mathbf{event} \ e_B(\text{sid}(\widetilde{msg}_{B,j}), k_{B,j}); \overline{c'_{B,j}[\tilde{i}'_j]} \langle M_{B,j} \rangle; (Q_{1B,j} \mid Q'_{2B,j}\{k_{B,j}/k, \text{sid}(\widetilde{msg}_{B,j})/x\}))_{1 \leq j \leq J}] \\ S_{\text{composed}} &= C''_h[Q_{\text{composed}}] \end{aligned}$$

Let  $S_{\text{composed}}^\circ$  be obtained from  $S_{\text{composed}}$  by removing all events of  $S_1$ .

Then we have the same properties as in Theorem 2 with  $\sum_{j=1}^J \prod \tilde{n}'_j$  instead of  $\prod \tilde{n}'$  and  $C''$  independent of  $Q_{1A}$  and  $Q_{1B,j}$  ( $1 \leq j \leq J$ ).

*Proof.* The proof is the same as for Theorem 2. We just need to repeat the treatment of the hole that contains event  $e_B$  for each  $j \in \{1, \dots, J\}$ ; the definition of  $\text{fresh}(\tilde{i}'', \tilde{u})$  is updated into  $\text{fresh}(\tilde{i}'', (\tilde{u}_j)_{1 \leq j \leq J}) = \mathbf{find} \ \bigoplus_{j=1}^J \tilde{u}'_j = \tilde{i}'''_j \leq \tilde{n}'_j \ \mathbf{suchthat} \ \mathbf{defined}(\tilde{u}_j[\tilde{i}''']) \wedge \tilde{u}_j[\tilde{i}'''] = \tilde{i}'' \ \mathbf{then} \ \mathbf{false} \ \mathbf{else} \ \mathbf{true}$ , using a **find** with several branches, so that we have  $\text{fresh}(\tilde{i}'', (\tilde{u}_j)_{1 \leq j \leq J})$  when  $\tilde{i}''$  was not used before, that is, it does not occur in any array  $\tilde{u}_j$  for  $j \in \{1, \dots, J\}$ .  $\square$

## A.11 Proof for Section 5.5

*Proof of Theorem 3.* The proof of this theorem is very similar to the proof of Theorem 2, so we do not repeat it, but just point out the differences. Let  $C_5$  be defined as in the proof of

Theorem 2 except that the condition  $\text{fresh}(\tilde{i}'', \tilde{u})$  is removed, because we do not have injectivity, and the process  $\text{Relay}(Q_{2B}, \tilde{i}', \tilde{u})$  is replaced with  $\text{Relay}(Q_{2B}, \tilde{i}', (\tilde{u}, \tilde{i}'))$ . Indeed, the indices of  $Q_{2B}$  are renumbered in a different way: the indices of  $Q_{2B}$  in the composed system are  $\tilde{i}'$ ; in Theorem 2, they are mapped to  $\tilde{u}[\tilde{i}']$  in  $S_2$ , while in this theorem, they are mapped to  $\tilde{u}[\tilde{i}'], \tilde{i}'$  in  $S_2$ . This change affects the rest of proof as we detail below. Let  $G_0$  be defined as in the proof of Theorem 2, except that the condition  $\text{fresh}(\tilde{i}'', \tilde{u})$  is removed. Let  $Q'_2$ ,  $Q'_{2B}$ , and  $Q'_2$  be defined as in the proof of Theorem 2. We have  $C_h''[C_5[Q'_2]] \approx_0^{V_1 \cup \{\tilde{u}\}} C_h''[C_5[Q'_2]]$ , as in Theorem 2.

Next, we show  $G_0 \xrightarrow{f', 0}^{V_1, V_1 \cup \{\tilde{u}\}} C_h''[C_5[Q'_2]]$ . To prove this property, we consider an evaluation context  $C_6$  acceptable for  $G_0$  with public variables  $V_1$ . Let  $f'(C_6) = C'_6$  be obtained from  $C_6$  by replacing array accesses  $y[\tilde{M}, \tilde{M}']$  with  $y[\tilde{u}[\tilde{M}], \tilde{M}, \tilde{M}']$ , when  $y \in V_1$  is defined in  $Q'_{2B}$ ,  $\tilde{M}$  contains as many elements as  $\tilde{i}'$ , and  $\tilde{M}'$  contains the other indices of  $y$  if any. We establish a correspondence between the traces of  $C_6[G_0]$  and those of  $C'_6[C_h''[C_5[Q'_2]]]$ : we eliminate communications on the private channels  $c'_1, c_2, \text{ch}(Q_{2B}), \text{ch}'(Q_{2B})$  (Appendix A.2) and we renumber the variables of  $Q'_{2B}$ , replacing indices  $\tilde{a} \leq \tilde{n}'$  in  $C_6[G_0]$  with  $\tilde{u}[\tilde{a}], \tilde{a} \leq \tilde{n}, \tilde{n}'$  in  $C'_6[C_h''[C_5[Q'_2]]]$ . In this correspondence, an execution of  $Q'_{2B} = \text{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\text{sid}}, Q_{2B})$  with replication indices  $\tilde{a}$  in  $C_6[G_0]$  corresponds to an execution of  $Q'_{2B} = \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_{2B})$  with replication indices  $\tilde{u}[\tilde{a}], \tilde{a}$  in  $C'_6[C_h''[C_5[Q'_2]]]$ . The proofs of  $G_0 \xrightarrow{f', 0}^{V_1, V_1 \cup \{\tilde{u}\}} C_h''[C_5[Q'_2]]$  and of  $S_{\text{composed}}^{\circ} \xrightarrow{f, p_3}^{V_1, V_2} S_2$  then proceed as in the proof of Theorem 2: the removal of the condition  $\text{fresh}$  compensates for the non-injectivity of the correspondence (7).

For  $y \in V_2 \cap \text{var}(Q_{2A})$ ,  $f$  is secrecy-preserving for  $y \mapsto (y, f_{\text{sec}})$  where  $f_{\text{sec}}(C_3) = f'(C_3)[C'_5[]]$ , by Lemma 10, as in Theorem 2. For  $y \in V_2 \cap \text{var}(Q_{2B})$ , the proof needs to be adapted since the renumbering of variables has changed. Suppose that  $S_2$  preserves the secrecy of  $y$  with public variables  $V_2 \setminus \{y\}$  up to probability  $p_2$ . By Lemma 1,  $C'_5[S_2]$  preserves the secrecy of  $y$  with public variables  $(V' \cup \text{var}(C'_5)) \setminus \{y\}$  up to probability  $p'_2$  such that  $p'_2(C_5) = p_1(C_5[C'_5])$ . Let  $R_y$  be the process used for testing secrecy of  $y$  in  $S_{\text{composed}}^{\circ}$ . Let  $R'_y$  be obtained by renumbering the variables  $R_y$ :  $R'_y = f'(R_y)$ , that is,

$$\begin{aligned} R'_y &= c_{s0}(); \mathbf{new} \ b : \text{bool}; \overline{c_{s0}}\langle \rangle; \\ &(!^{i_s \leq n_s} c_s[i_s](\tilde{u}_1 \leq \tilde{n}', \tilde{u}_2 \leq \tilde{n}_2); \mathbf{if} \ \text{defined}(y[\tilde{u}[\tilde{u}_1], \tilde{u}_1, \tilde{u}_2]) \ \mathbf{then} \\ &\quad \mathbf{if} \ b \ \mathbf{then} \ \overline{c_s[i_s]}\langle y[\tilde{u}[\tilde{u}_1], \tilde{u}_1, \tilde{u}_2] \rangle \ \mathbf{else} \\ &\quad \mathbf{find} \ u'_s = i'_s \leq n_s \ \mathbf{suchthat} \ \text{defined}(y'[i'_s], \tilde{u}_1[i'_s], \tilde{u}_2[i'_s]) \wedge \tilde{u}_1[i'_s] = \tilde{u}_1 \wedge \tilde{u}_2[i'_s] = \tilde{u}_2 \\ &\quad \mathbf{then} \ \overline{c_s[i_s]}\langle y'[u'_s] \rangle \ \mathbf{else} \ \mathbf{new} \ y' : T; \overline{c_s[i_s]}\langle y' \rangle \\ &| c'_s(b'); \mathbf{if} \ b = b' \ \mathbf{then} \ \mathbf{event\_abort} \ S \ \mathbf{else} \ \mathbf{event\_abort} \ \bar{S}) \end{aligned}$$

Let  $c'_s$  be a fresh channel,  $\tilde{u}'_1, \tilde{u}'_2, y'', \tilde{u}''_1, \tilde{u}''_2, y'''$  be fresh variables, and let us define relay processes:

$$\begin{aligned} \text{Relay}_R &= !^{i_s \leq n_s} c'_s[i_s](\tilde{u}'_1 \leq \tilde{n}', \tilde{u}'_2 \leq \tilde{n}_2); \mathbf{if} \ \text{defined}(\tilde{u}[\tilde{u}'_1]) \ \mathbf{then} \\ &\quad \overline{c_s[i_s]}\langle \tilde{u}[\tilde{u}'_1], \tilde{u}'_1, \tilde{u}'_2 \rangle; c_s[i_s](y'' : T); \overline{c'_s[i_s]}\langle y'' \rangle \\ \text{Relay}'_R &= !^{i_s \leq n_s} c_s[i_s](\tilde{u}''_1 \leq \tilde{n}', \tilde{u}''_2 \leq \tilde{n}_2); \overline{c'_s[i_s]}\langle \tilde{u}''_1, \tilde{u}''_2 \rangle; c'_s[i_s](y''' : T); \overline{c_s[i_s]}\langle y''' \rangle \end{aligned}$$

The process  $\text{Relay}_R$  renumbers the queries for  $y$  like  $f'$  renumbers  $y$ . However, it uses channel  $c'_s$  instead of  $c_s$ , so we use process  $\text{Relay}'_R$  to revert to channel  $c_s$ . Let  $C_R = \mathbf{newChannel} \ c'_s; (\text{Relay}'_R \mid \mathbf{newChannel} \ c_s; (\text{Relay}_R \mid []))$ . Let  $R''_y$  be the process used for testing secrecy of  $y$  in  $C'_5[S_2]$ . (The process  $R''_y$  differs from  $R_y$  because  $y$  has indices  $\tilde{i}' \leq \tilde{n}', \tilde{i}_2 \leq \tilde{n}_2$  in  $S_{\text{composed}}^{\circ}$ ,

while it has indices  $\tilde{i} \leq \tilde{n}, \tilde{i}' \leq \tilde{n}', \tilde{i}_2 \leq \tilde{n}_2$  in  $C'_5[S_2]$ .) We have

$$C'_5[S_2] \mid R'_y \approx_0^{V_1} C_R[C'_5[S_2] \mid R''_y]$$

by eliminating communications on the private channels  $c_s$  and  $c''_s$  (Appendix A.2). The equality test  $\tilde{u}_1[i'_s] = \tilde{u}_1$  performed by  $R'_y$  in the left-hand side becomes  $\tilde{u}[\tilde{u}_1[i'_s]] = \tilde{u}[\tilde{u}_1] \wedge \tilde{u}_1[i'_s] = \tilde{u}_1$  in the right-hand side because  $R''_y$  always receives  $\tilde{u}[\tilde{u}_1], \tilde{u}_1$  instead of  $\tilde{u}_1$ . It is easy to see that these two tests are equivalent: the second test includes the first one as a conjunct, and conversely when  $\tilde{u}_1[i'_s] = \tilde{u}_1$  holds, we obviously have  $\tilde{u}[\tilde{u}_1[i'_s]] = \tilde{u}[\tilde{u}_1]$ . The rest of the proof that  $f$  is secrecy-preserving for  $y \mapsto (y, f_{\text{sec}})$  proceeds exactly as in Theorem 2.

The proof of the second point of the theorem also proceeds as in Theorem 2.  $\square$

## A.12 Single Process with Key Reuse

The next theorem is a weakened variant of Theorem 1 that allows us to reuse the same key  $k$  several times. This situation complicates the theorem considerably.

**Theorem 5.** *Let  $C$  be any context with one hole, with replications  $!^{\tilde{i} \leq \tilde{n}}$  above the hole and without **event\_abort**. Let  $Q_1$  be a process without **event\_abort**. Let  $M$  be a term of type  $T$ . Let*

$$\begin{aligned} Q'_1 &= C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\ &\quad \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(\widetilde{msg}[\tilde{i}'], k_1[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ &\quad \text{then } \overline{c_1[\tilde{i}]} \langle M_1 \rangle; Q_1 \\ &\quad \text{else let } k_1 = k \text{ in } \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; Q_1] \\ Q_2 &= !^{\tilde{i}'' \leq \tilde{n}} Q_0 \\ Q'_2 &= c_2(); \text{new } k : T; \overline{c_3} \langle \rangle; Q_2 \\ S_1 &= C_h[Q'_1] \\ S_2 &= C'_h[\text{AddReplSid}(\tilde{i} \leq \tilde{n}, c'_2, T_{\text{sid}}, Q'_2)] \end{aligned}$$

where  $Q'_1$  and  $Q'_2$  are hash-well-formed;  $\widetilde{msg}$  is a sequence of variables input or output by  $C$  above the hole;  $\text{sid}$  is a function that takes a sequence of messages and returns a session identifier of type  $T_{\text{sid}}$ ;  $c_1, c'_1, c_2, c'_2, c_3, e, k_1, \tilde{u}$  do not occur elsewhere in  $S_1, S_2$ ;  $k$  does not occur elsewhere in  $S_1$  ( $k$  occurs in  $Q_0$ );  $k$  is the only common variable between  $S_1$  and  $S_2$ ;  $S_1$  and  $S_2$  have no common channel, no common event, and no common table;  $S_1$  and  $S_2$  do not contain **newChannel**;  $Q_2$  contains no **defined** condition; and  $C$  and  $Q_2$  make all their inputs and outputs on pairwise distinct channels with indices the current replication indices. Let  $\text{ch}(Q_0)$  be the channels of  $Q_0$ . For each channel  $c$  in  $\text{ch}(Q_0)$ , let  $\text{fr}_c$  be a fresh channel.

Let  $Q'_0 = \text{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\text{sid}}, Q_0)$ . Let  $c'_1$  be a fresh channel. Let

$$\begin{aligned} Q_{\text{composed}} &= C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; \\ &\quad (Q_1 \mid Q'_0\{\tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})] \\ S_{\text{composed}} &= C''_h[Q_{\text{composed}}] \end{aligned}$$

Let  $S_{\text{composed}}^\circ$  be obtained from  $S_{\text{composed}}$  by removing the events of  $S_1$ .

Let  $t_1 = t_C + \prod \tilde{n} \times (t_M + t_{Q_1})$  be an upper bound on the runtime of  $Q'_1$ ,  $t_2 = \prod \tilde{n} \times t_{Q_0}$  be an upper bound on the runtime of  $Q'_0$  in  $Q_{\text{composed}}$ ,  $n_{h,l,1} = n_{h,l,C} + \prod \tilde{n} \times (n_{h,l,M} + n_{h,l,Q_1})$ , and  $n_{h,l,2} = \prod \tilde{n} \times n_{h,l,Q_0}$ .

1. If  $S_1$  preserves the secrecy of  $k_1$  with public variables  $V$  ( $V \subseteq \text{var}(S_1) \setminus (\{k_1\} \cup \text{var}(C_h))$ ) up to probability  $p$  and  $S_1$  satisfies the correspondence  $\text{event}(e(\text{sid}, k)) \wedge \text{event}(e(\text{sid}, k')) \implies k = k'$  with public variables  $V \cup \{k_1\}$  up to probability  $p'$ , then there exists an evaluation context  $C_5^\circ$  such that, for any  $V_1 \subseteq V \cup (\text{var}(Q_2') \setminus (\{k\} \cup \text{var}(C_h')))$ , we have  $S_{\text{composed}}^\circ \approx_{p''}^{V_1} C_5^\circ[S_2]$ ,  $C_5^\circ$  is acceptable for  $S_2$  without public variables and contains no event, and  $C_5^\circ$  runs in time at most  $t_1$ , calls the  $l$ -th hash oracle at most  $n_{h,l,1}$  times, so  $n'_{h,l} = n''_{h,l} + n_{h,l,1}$ , and does not alter the other parameters, where  $p''(C_3, t_D) = p(C'_3 + t_D) + p'(C'_3, t_D)$  and, assuming  $C_3$  calls the  $l$ -th hash oracle at most  $n''_{h,l}$  times, the context  $C'_3$  runs in time at most  $t_{C_3} + t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l} = n''_{h,l} + n_{h,l,2}$  times, and its other parameters are the same as those of  $C_3$ .
2. There exists an evaluation context  $C'_4$  such that, for any  $V' \subseteq \text{var}(S_{\text{composed}}) \setminus (\{k_1, \tilde{u}\} \cup \text{var}(C_h''))$ , we have  $S_{\text{composed}} \approx_0^{V'} C'_4[S_1]$  and  $C'_4$  is acceptable for  $S_1$  with public variable  $k$ , contains the events of  $S_2$ , runs in time at most  $t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l,2}$  times, so  $n_{h,l} = n''_{h,l} + n_{h,l,2}$ , and does not alter the other parameters.

Moreover,  $C_5^\circ$  is independent of the details of  $Q_0$ : it depends only on the channels of  $Q_0$ , whether they are for input or for output, under which replications and with which type of data;  $C'_4$  is independent of  $Q_1$ .

In this theorem, the system  $S_1$  establishes a key  $k$  and the system  $S_2$  creates a fresh key  $k$  and runs several sessions of  $Q_0$  using this key. The composed system runs  $S_1$  and  $Q_0$  using the key  $k$  provided by  $S_1$ . As in Theorem 2, these systems can share hash oracles, included in  $C_h$ ,  $C'_h$ , and  $C''_h$ .

Inside  $S_1$ , the **find** and the assignment  $k_1 = k$  in  $Q'_1$  just serve for specifying the desired security properties of  $S_1$ , as follows. When the session identifier  $\text{sid}(\widetilde{msg})$  has not been seen before,  $S_1$  stores its key  $k$  in  $k_1$ , and we require that  $k_1$  be secret. As a consequence of secrecy, the keys obtained with different session identifiers are indistinguishable from independent random numbers. Furthermore, the correspondence  $\text{event}(e(\text{sid}, k)) \wedge \text{event}(e(\text{sid}, k')) \implies k = k'$  guarantees that, when another session has the same session identifier  $\text{sid}$ , then it also has the same key. Since the session identifiers can be computed from public messages  $\widetilde{msg}$ , the adversary knows whether  $S_1$  reuses the same key or uses a fresh key. Therefore, a context around  $S_1$  can know whether it should run another session of  $Q_0$  with the same key or a new session of  $Q_0$  with a fresh key. This observation is important for the proof of the first point of Theorem 5.

As in the previous theorems, the first point of Theorem 5 allows us to transfer security properties from  $S_2$  to the composed system  $S_{\text{composed}}$ , and the second point allows us to transfer security properties from  $S_1$  to  $S_{\text{composed}}$ .

*Proof of Theorem 5.* We use the notations of relay processes of the proof of Theorem 2. Let  $k'$  and  $r$  be fresh variables. Let

$$\begin{aligned}
& C_5 = \text{newChannel } c'_2, c_3, \text{ch}(Q_0); \\
& (C[\text{let } k' = M \text{ in event } e(\text{sid}(\widetilde{msg}), k'); \\
& \quad \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(r[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\
& \quad \text{then } \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{u}, \tilde{i}))) \\
& \quad \text{else let } r = \text{cst in } \overline{c'_2[\tilde{i}]} \langle \text{sid}(\widetilde{msg}) \rangle; c_3[\tilde{i}](); \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{i}, \tilde{i}))) \\
& \quad \mid []])
\end{aligned}$$

In the **then** branch, the key  $k$  is the same as for indices  $\tilde{u}$ , so we reuse it. In the **else** branch, a new key  $k$  is created by sending a message on  $c'_2[\tilde{i}]$  and receiving the reply on  $c_3[\tilde{i}]$ .

Let  $Q_2'' = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c_2', T_{\text{sid}}, Q_2')$ , so that we have  $S_2 = C_h'[Q_2'']$ . By Lemma 4, there exists an evaluation context  $C_5'$  such that

$$C_h''[C_5[Q_2'']] \approx_0^{V_1} C_5'[C_h'[Q_2'']] = C_5'[S_2] \quad (21)$$

where the context  $C_5'$  runs in time at most  $t_{C_5} \leq t_1$ , calls the  $l$ -th hash oracle in  $C_h'$  at most  $n_{h,l,C_5} \leq n_{h,l,1}$  times, so  $n'_{h,l} = n''_{h,l} + n_{h,l,1}$ , and its other parameters are the same as those of  $C_5$ .

Let

$$\begin{aligned} G_1 = C_h''[\text{newChannel } \text{ch}(Q_0); C[ \\ & \text{let } k' = M \text{ in event } e(\text{sid}(\widetilde{msg}), k'); \\ & \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(r[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ & \text{then } \overline{c_1''[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{u}, \tilde{i}))) \\ & \text{else} \\ & \text{let } r = \text{cst in new } k : T; \overline{c_1''[\tilde{i}]} \langle M_1 \rangle; \\ & (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{i}, \tilde{i})) \mid \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_2) \{ \text{sid}(\widetilde{msg})/x \})] \end{aligned}$$

We have  $C_h''[C_5[Q_2'']] \approx_0^{V_1} G_1$  by eliminating communications on  $c_2'$  and  $c_3$  (Appendix A.2). Note that the **find** introduced at the root of  $Q_2''$  by **AddReplSid** never succeeds because the session identifier  $\text{sid}(\widetilde{msg})$  is sent on  $c_2'$  only when it was not seen before. Let

$$\begin{aligned} G_2 = C_h''[\text{newChannel } \text{ch}(Q_0); C[ \\ & \text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\ & \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(k_1[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ & \text{then } \overline{c_1''[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{u}, \tilde{i}))) \\ & \text{else} \\ & \text{new } k_1 : T; \overline{c_1''[\tilde{i}]} \langle M_1 \rangle; \\ & (Q_1 \mid \text{Relay}(Q_0, \tilde{i}, (\tilde{i}, \tilde{i})) \mid \text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_2) \{ k_1/k, \text{sid}(\widetilde{msg})/x \})] \end{aligned}$$

We also have  $G_1 \approx_0^{V_1} G_2$  by renaming  $k$  into  $k_1$  and  $k'$  into  $k$ , and using the definition of  $k_1[\tilde{i}']$  instead of the definition of  $r[\tilde{i}']$ , which is equivalent since these variables are defined at the same time. That allows us to remove the variable  $r$ . Therefore, by transitivity,  $C_5'[S_2] \approx_0^{V_1} G_2$ . Let  $C_5'^\circ$  and  $G_2^\circ$  be obtained from  $C_5'$  and  $G_2$  respectively by removing the events of  $S_1$ . By Lemma 11, we have

$$C_5'^\circ[S_2] \approx_0^{V_1} G_2^\circ \quad (22)$$

Let

$$\begin{aligned} Q_1'' = C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\ & \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(k_1[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ & \text{then } \overline{c_1[\tilde{i}]} \langle M_1 \rangle; Q_1 \\ & \text{else new } k_1 : T; \overline{c_1[\tilde{i}]} \langle (k_1, M_1) \rangle; Q_1] \end{aligned}$$

Let  $G_3$  be obtained by replacing **new**  $k_1 : T$  with **let**  $k_1 = k$  **in** in  $G_2$ . Let  $Q_1'''$  be obtained by replacing **new**  $k_1 : T$  with **let**  $k_1 = k$  **in** in  $Q_1''$ . Let  $G_3^\circ$ ,  $Q_1''^\circ$ , and  $Q_1'''^\circ$  be obtained from  $G_3$ ,  $Q_1''$ , and  $Q_1'''$  respectively by removing the events of  $S_1$ .

By introducing a relay process and renaming channels as in the proof of Theorem 2, we have

$$G_2^\circ \{ \text{fr}_c/c, c \in \text{ch}(C) \} \approx_0^{V_1} C_h''[C_1[Q_1''^\circ]] \quad (23)$$

$$G_3^\circ \{ \text{fr}_c/c, c \in \text{ch}(C) \} \approx_0^{V_1} C_h''[C_1[Q_1'''^\circ]] \quad (24)$$

for some evaluation context  $C_1$  that runs in time at most  $t_2$ . By Lemma 4, there exists an evaluation context  $C_1'$  such that

$$C_h''[C_1[Q_1''^\circ]] \approx_0^{V_1} C_1'[C_h[Q_1''^\circ]] \quad (25)$$

$$C_h''[C_1[Q_1'''^\circ]] \approx_0^{V_1} C_1'[C_h[Q_1'''^\circ]] \quad (26)$$

where the context  $C_1'$  runs in time at most  $t_{C_1} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_1} \leq n_{h,l,2}$  times, so  $n_{h,l} = n_{h,l}' + n_{h,l,2}$ , and its other parameters are the same as those of  $C_1$ .

Since  $S_1 = C_h[Q_1']$  preserves the secrecy of  $k_1$  with public variables  $V$  ( $k_1, k \notin V$ ) up to probability  $p$ , by Lemma 9, we have  $C_h[Q_1''^\circ] \approx_{p_0}^V C_h[Q_1'''^\circ]$  where  $p_0(C_3, t_D) = p(C_3 + t_D)$ . Therefore,

$$C_1'[C_h[Q_1''^\circ]] \approx_{p_1}^{V_1} C_1'[C_h[Q_1'''^\circ]] \quad (27)$$

where  $p_1(C_3, t_D) = p_0(C_3[C_1'], t_D) = p(C_3[C_1'] + t_D) = p(C_3' + t_D)$  and the context  $C_3'$  runs in time at most  $t_{C_3} + t_2$ , calls the  $l$ -th hash oracle at most  $n_{h,l} = n_{h,l}' + n_{h,l,2}$  times, and its other parameters are the same as those of  $C_3$ . So by combining (23), (25), (27), (26), and (24),

$$G_2^\circ \{ \text{fr}_c/c, c \in \text{ch}(C) \} \approx_{p_1}^{V_1} G_3^\circ \{ \text{fr}_c/c, c \in \text{ch}(C) \}$$

so by renaming channels

$$G_2^\circ \approx_{p_1}^{V_1} G_3^\circ \quad (28)$$

Let

$$\begin{aligned} G_4 &= C_h''[C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\ &\quad \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(k_1[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ &\quad \text{then } (*) \overline{c_1'[\tilde{i}]}(M_1); (Q_1 \mid Q_0'\{k_1[\tilde{u}]/k, \tilde{i}/\tilde{i}', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\}) \\ &\quad \text{else let } k_1 = k \text{ in } \overline{c_1'[\tilde{i}]}(M_1); (Q_1 \mid Q_0'\{k_1/k, \tilde{i}/\tilde{i}', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})]] \end{aligned}$$

We have

$$G_3 \approx_0^{V_1} G_4$$

by eliminating communications on  $\text{ch}(Q_0)$  (Appendix A.2) and mapping all variables  $x[\tilde{b}, \tilde{a}, \dots]$  ( $\tilde{b} \leq \tilde{n}, \tilde{a} \leq \tilde{n}$ ) of  $\text{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\text{sid}}, Q_2)\{k_1/k, \text{sid}(\widetilde{msg})/x\}$  to  $x[\tilde{a}, \dots]$ , which is possible without clashes since there is a single value of  $\tilde{b}$  actually used for each  $\tilde{a}$  ( $\tilde{b} = \tilde{u}[\tilde{a}]$  when  $\tilde{u}[\tilde{a}]$  is defined;  $\tilde{b} = \tilde{a}$  otherwise). The table entries and the events are the same on both sides because they contain the session identifier  $x = \text{sid}(\widetilde{msg})$ , not the replication indices  $\tilde{i}$ . By introducing again a relay process and renaming channels as in the proof of Theorem 2, we have  $G_4\{\text{fr}_c/c, c \in \text{ch}(C)\} \approx_0^{V_1} C_h''[C_2[Q_1']]$  for some evaluation context  $C_2$  that runs in time at most  $t_2$  and uses the public variable  $k_1$ . By Lemma 4, there exists an evaluation context  $C_2'$  such that  $C_h''[C_2[Q_1']] \approx_0^{V_1} C_2'[C_h[Q_1']] = C_2'[S_1]$  where the context  $C_2'$  runs in time at most  $t_{C_2} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_2} \leq n_{h,l,2}$  times, so  $n_{h,l} = n_{h,l}' + n_{h,l,2}$ , and its other parameters are the same as those of  $C_2$ .



Since  $S_1$  satisfies the correspondence  $\mathbf{event}(e(sid, k)) \wedge \mathbf{event}(e(sid, k')) \implies k = k'$  with public variables  $V \cup \{k_1\}$  up to probability  $p'$ , by Lemma 1,  $C'_2[S_1]$  satisfies this correspondence with public variables  $V_1$  up to probability  $p'_1(C_3, t_D) = p'(C_3[C'_2], t_D) = p'(C'_3, t_D)$  where the context  $C'_3$  runs in time at most  $t_{C_3} + t_2$  and its other parameters are the same as those of  $C_3$ , and so do  $C''_h[C_2[Q'_1]]$  and  $G_4$ . Therefore, we infer that, at the program point  $(*)$  in  $G_4$ ,  $k_1[\tilde{u}] = k[\tilde{u}] = k$  except in cases of probability  $p'_1$ , since  $\text{sid}(\widetilde{msg}[\tilde{u}]) = \text{sid}(\widetilde{msg})$ . So we have  $G_4 \approx_{p'_1}^{V_1} G_5$  where

$$\begin{aligned} G_5 = & C''_h[C[\mathbf{let} \ k = M \ \mathbf{in} \ \mathbf{event} \ e(\text{sid}(\widetilde{msg}), k); \\ & \mathbf{find} \ \tilde{u} = \tilde{i}' \leq \tilde{n} \ \mathbf{suchthat} \ \mathbf{defined}(k_1[\tilde{i}'], \widetilde{msg}[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\ & \mathbf{then} \ \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{\text{sid}(\widetilde{msg})/x, \tilde{i}/\tilde{i}'', \text{fr}_c/c, c \in \text{ch}(Q_0)\}) \\ & \mathbf{else} \ \mathbf{let} \ k_1 = k \ \mathbf{in} \ \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{\text{sid}(\widetilde{msg})/x, \tilde{i}/\tilde{i}'', \text{fr}_c/c, c \in \text{ch}(Q_0)\})]] \end{aligned}$$

Moreover,  $G_5 \approx_0^{V_1} S_{composed}$  since  $k_1$  is used only in the test of **find** ( $k_1 \notin V$ ) and both branches of **find** execute the same code except for the assignment to  $k_1$ , so the **find** and the assignment to  $k_1$  can be removed. Therefore, by transitivity,  $G_3 \approx_{p'_1}^{V_1} S_{composed}$ , so by Lemma 11,

$$G_3^\circ \approx_{p'_1}^{V_1} S_{composed}^\circ \quad (29)$$

By combining (22), (28), and (29) by transitivity, we obtain

$$C'_5 \circ [S_2] \approx_{p_1+p'_1}^{V_1} S_{composed}^\circ$$

Let us prove the second point. Let  $T_1$  be the type of  $M_1$ . Let  $k'$  be a fresh variable not in  $V'$ . Let  $\widetilde{msg}'$  be the sequence of variables corresponding to  $\widetilde{msg}$ , but using variables of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  instead of variables of  $C$ . As above, the goal of the relay process  $\text{Relay}(C, \emptyset, \emptyset)$  is to capture in variables  $\widetilde{msg}'$ , visible outside  $C$ , the content of  $\widetilde{msg}$ , which are variables internal to  $C$  but sent or received on public channels. Since  $\text{Relay}(C, \emptyset, \emptyset)$  renames channels  $c \in \text{ch}(C)$  to  $\text{fr}_c$ , we use the relay process  $\text{Relay}'(C, \emptyset)$  to perform the reverse renaming.

Let

$$\begin{aligned} C_4 = & \mathbf{newChannel} \ \text{ch}'(C); (\text{Relay}'(C, \emptyset) \mid \\ & \mathbf{newChannel} \ c_1, c'_1, \text{ch}(C); (\text{Relay}(C, \emptyset, \emptyset) \mid \\ & \tilde{i}' \leq \tilde{n} \ c_1[\tilde{i}](y_1 : T_1); \mathbf{if} \ \mathbf{defined}(k[\tilde{i}], \widetilde{msg}'[\tilde{i}]) \ \mathbf{then} \\ & \quad \mathbf{let} \ k' = k[\tilde{i}] \ \mathbf{in} \ \overline{c'_1[\tilde{i}]} \langle y_1 \rangle; Q'_0\{k'/k, \tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg}'[\tilde{i}])/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\} \mid \\ & \tilde{i}' \leq \tilde{n} \ c'_1[\tilde{i}](y_2 : T_1); \mathbf{if} \ \mathbf{defined}(k[\tilde{i}], \widetilde{msg}'[\tilde{i}]) \ \mathbf{then} \\ & \quad \mathbf{let} \ k' = k[\tilde{i}] \ \mathbf{in} \ \overline{c'_1[\tilde{i}]} \langle y_2 \rangle; Q'_0\{k'/k, \tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg}'[\tilde{i}])/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\} \mid \\ & [])) \end{aligned}$$

By Lemma 4, there exists an evaluation context  $C'_4$  such that

$$C''_h[C_4[Q'_1]] \approx_0^{V'} C'_4[C_h[Q'_1]] = C'_4[S_1]$$

where the context  $C'_4$  runs in time at most  $t_{C_4} \leq t_2$ , calls the  $l$ -th hash oracle in  $C_h$  at most  $n_{h,l,C_4} \leq n_{h,l,2}$  times, so  $n_{h,l} = n''_{h,l} + n_{h,l,2}$ , and its other parameters are the same as those of  $C_4$ , that is, it does not alter the other parameters.

Hence, we have

$$\begin{aligned}
C'_4[S_1] &\approx_0^{V'} \\
C''_h[C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\
&\quad \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(\widetilde{msg}[\tilde{i}'], k_1[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\
&\quad \text{then} \\
&\quad \quad (\text{if defined}(k[\tilde{i}]) \text{ then let } k' = k[\tilde{i}] \text{ in} \\
&\quad \quad \quad \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{k'/k, \tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})) \\
&\quad \text{else} \\
&\quad \quad \text{let } k_1 = k \text{ in if defined}(k[\tilde{i}]) \text{ then} \\
&\quad \quad \text{let } k' = k[\tilde{i}] \text{ in } \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{k'/k, \tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})]]]
\end{aligned}$$

by eliminating communications on the private channels  $\text{ch}'(C)$ ,  $c_1$ ,  $c'_1$ ,  $\text{ch}(C)$  (Appendix A.2), so

$$\begin{aligned}
C'_4[S_1] &\approx_0^{V'} C''_h[C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\
&\quad \text{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \text{ suchthat defined}(\widetilde{msg}[\tilde{i}'], k_1[\tilde{i}']) \wedge \text{sid}(\widetilde{msg}[\tilde{i}']) = \text{sid}(\widetilde{msg}) \\
&\quad \text{then } \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{\tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\}) \\
&\quad \text{else let } k_1 = k \text{ in } \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{\tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})]]]
\end{aligned}$$

since  $k$  is an abbreviation for  $k[\tilde{i}]$ , so  $k[\tilde{i}]$  is always defined and  $k' = k$ . Hence

$$\begin{aligned}
C'_4[S_1] &\approx_0^{V'} C''_h[C[\text{let } k = M \text{ in event } e(\text{sid}(\widetilde{msg}), k); \\
&\quad \overline{c'_1[\tilde{i}]} \langle M_1 \rangle; (Q_1 \mid Q'_0\{\tilde{i}/\tilde{i}'', \text{sid}(\widetilde{msg})/x, \text{fr}_c/c, c \in \text{ch}(Q_0)\})]] \\
&= S_{\text{composed}}
\end{aligned}$$

since both branches of **find** execute the same code except for the assignment to  $k_1$  and  $k_1$  and  $\tilde{u}$  are not used ( $\{k_1, \tilde{u}\} \cap V' = \emptyset$ ).  $\square$

### A.13 Lemma on a Process that Chooses a Bit

In this section, we consider processes  $P_0$  and  $P_1$  that do not contain  $b_0$  and a process

$$Q = c(x : T); \text{new } b_0 : \text{bool}; \text{if } b_0 \text{ then } P_1 \text{ else } P_0$$

that chooses a random bit  $b_0$  and runs  $P_0$  or  $P_1$  depending on the value of this bit. For  $v \in \{0, 1\}$ , we define  $Q_v = c(x : T); P_v$ . The process  $Q_v$  runs as  $Q$  when  $b_0 = v$ . (0 stands for false and 1 for true.) Let  $Q_v^\circ$  be obtained from  $Q_v$  by removing all events. We have the following lemma.

**Lemma 12.** *If  $Q$  preserves the secrecy of  $b_0$  with public variables  $V$  up to probability  $p$  and  $Q$  does not contain **event\_abort**, then  $Q_0^\circ \approx_p^{V'} Q_1^\circ$  where  $p'(C, t_D) = 4p(C + t_D)$ .*

Intuitively, the adversary can distinguish  $Q_0^\circ$  from  $Q_1^\circ$  if and only if it can determine the value of  $b_0$ , that is,  $b_0$  is not secret.

*Proof.* Let  $C$  be any acceptable evaluation context for  $Q_0$  and  $Q_1$  with public variables  $V$ , and  $D$  a distinguisher. Let  $c_{s0}$ ,  $c_s$ , and  $c'_s$  be channels that  $C$  does not use.

Let  $C'$  be a context that runs  $C$  but stores events executed by  $C$  in its internal state instead of actually executing events, computes  $D$  on the stored sequence of events executed by  $C$  and stores the result in  $b'_0$ , outputs on channel  $c_{s0}$  and inputs on channel  $c_{s0}$ , outputs on channel  $c_s$ , receives the answer  $b''_0$  on channel  $c_s$ . If  $b'_0 = b''_0$ , it chooses a random  $b'$  and sends it on channel  $c'_s$ . If  $b'_0 \neq b''_0$ , it sends **false** on channel  $c'_s$ . Such a context  $C'$  exists because it can be encoded as a probabilistic Turing machine adversary, which can itself be encoded as a context in CryptoVerif [14, Section 2.8].

When  $b_0 = v$ ,  $C'[Q \mid R_x]$  stores in  $b'_0$  the result of  $C[Q_v^\circ] : D$ . When  $b'_0 = v$ ,

- if  $b = \text{true}$  (probability  $1/2$ ), then  $b''_0 = b_0 = v$ , so  $b'$  is random:  $b' = \text{true}$  and  $\mathbf{S}$  is executed with probability  $1/4$  and  $b' = \text{false}$  and  $\bar{\mathbf{S}}$  is executed with probability  $1/4$ ;
- if  $b = \text{false}$  (probability  $1/2$ ), then  $b''_0$  is random, so
  - $b''_0 = v$  with probability  $1/4$ , so  $b'$  is random:  $b' = \text{true}$  and  $\bar{\mathbf{S}}$  is executed with probability  $1/8$  and  $b' = \text{false}$  and  $\mathbf{S}$  is executed with probability  $1/8$ ;
  - $b''_0 \neq v$  with probability  $1/4$ , and in this case  $b' = \text{false}$  and  $\mathbf{S}$  is executed.

When  $b'_0 \neq v$ ,

- if  $b = \text{true}$  (probability  $1/2$ ), then  $b''_0 = b_0 = v$ , so  $b' = \text{false}$  and  $\bar{\mathbf{S}}$  is executed with probability  $1/2$ ;
- if  $b = \text{false}$  (probability  $1/2$ ), then  $b''_0$  is random, so
  - $b''_0 \neq v$  with probability  $1/4$ , so  $b'$  is random:  $b' = \text{true}$  and  $\bar{\mathbf{S}}$  is executed with probability  $1/8$  and  $b' = \text{false}$  and  $\mathbf{S}$  is executed with probability  $1/8$ ;
  - $b''_0 = v$  with probability  $1/4$ , and in this case  $b' = \text{false}$  and  $\mathbf{S}$  is executed.

So

$$\begin{aligned} \Pr[C'[Q \mid R_x] : \mathbf{S} / b_0 = v] &= \frac{5}{8} \Pr[C[Q_v^\circ] : D = v] + \frac{3}{8} \Pr[C[Q_v^\circ] : D \neq v] \\ &= \frac{3}{8} + \frac{1}{4} \Pr[C[Q_v^\circ] : D = v] \end{aligned}$$

because  $\Pr[C[Q_v^\circ] : D \neq v] = 1 - \Pr[C[Q_v^\circ] : D = v]$ . Finally, since  $C'[Q \mid R_x]$  always executes either  $\mathbf{S}$  or  $\bar{\mathbf{S}}$ , we obtain

$$\begin{aligned} &\Pr[C'[Q \mid R_x] : \mathbf{S}] - \Pr[C'[Q \mid R_x] : \bar{\mathbf{S}}] \\ &= 2 \cdot \Pr[C'[Q \mid R_x] : \mathbf{S}] - 1 \\ &= \Pr[C'[Q \mid R_x] : \mathbf{S} / b_0 = 0] + \Pr[C'[Q \mid R_x] : \mathbf{S} / b_0 = 1] - 1 \\ &= \frac{3}{8} + \frac{1}{4} \Pr[C[Q_0^\circ] : D = 0] + \frac{3}{8} + \frac{1}{4} \Pr[C[Q_1^\circ] : D = 1] - 1 \\ &= \frac{1}{4} (1 - \Pr[C[Q_0^\circ] : D]) + \frac{1}{4} \Pr[C[Q_1^\circ] : D] - \frac{1}{4} \\ &= \frac{1}{4} (\Pr[C[Q_1^\circ] : D] - \Pr[C[Q_0^\circ] : D]) \end{aligned}$$

so  $\Pr[C[Q_1^\circ] : D] - \Pr[C[Q_0^\circ] : D] = 4(\Pr[C'[Q \mid R_x] : \mathbf{S}] - \Pr[C'[Q \mid R_x] : \bar{\mathbf{S}}]) \leq 4p(C + t_D) = p'(C, t_D)$   $\square$

## B Details on the Proof of TLS

This appendix relies on the CryptoVerif analysis of TLS by Bhargavan et al. described in [10, Section 6]. The CryptoVerif scripts for their analysis are available at <https://github.com/Inria-Prosecco/reftls/tree/master/cv>. We provide a brief reminder of the results of this analysis, before explaining the composition. For the full details, we still recommend reading [10, Section 6] before this appendix.

### B.1 Reminder

Even though CryptoVerif can evaluate the probability of success of an attack as a function of the number of sessions and the probability of breaking each primitive (exact security), for simplicity, as in [10], we consider the asymptotic framework in which we only show that the probability of success of an attack is negligible as a function of the security parameter  $\eta$ . (A function  $f$  is *negligible* when for all polynomials  $q$ , there exists  $\eta_0 \in \mathbb{N}$  such that for all  $\eta > \eta_0$ ,  $f(\eta) \leq \frac{1}{q(\eta)}$ .) Therefore, we omit probabilities from the security properties. All processes run in polynomial time in the security parameter and manipulate bitstrings of polynomially bounded length.

The analysis of [10, Section 6] splits TLS 1.3 into three components: the initial handshake, the handshakes with pre-shared key, and the record protocol.

**Initial Handshake** The initial handshake is a Diffie-Hellman key-exchange protocol between a client and a server. It provides 4 keys: the server application traffic secret *sats* used by the record protocol for sending messages from the server to the client, the client application traffic secret *cats* similar for messages from the client to the server, the exporter master secret *ems*, and the resumption secret *resumption\_secret* used as pre-shared key by the next handshake. The key *sats* can be used to send messages from the server to the client before the last message of the handshake; these messages are called 0.5-RTT messages. These keys are stored in variables with prefix *c\_* on the client side and *s\_* on the server side. Furthermore, on the server side, the variables *s\_cats*, *s\_ems*, and *s\_resumption\_secret* are duplicated, with suffixes 1 and 2, for a technical reason (see [10]). CryptoVerif proves the following correspondences:

$$\text{inj-event}(\text{ClientTerm}(\log_4, s\_keys)) \implies \text{inj-event}(\text{ServerAccept}(\log_4, s\_keys, i)) \quad (30)$$

$$\text{event}(\text{ServerAccept}(\log_4, s\_keys, i)) \wedge \text{event}(\text{ServerAccept}(\log_4, s\_keys', i')) \implies i = i' \quad (31)$$

$$\text{inj-event}(\text{ServerTerm}(\log_7, c\_keys)) \implies \text{inj-event}(\text{ClientAccept}(\log_7, c\_keys, i)) \quad (32)$$

$$\text{event}(\text{ClientAccept}(\log_7, c\_keys, i)) \wedge \text{event}(\text{ClientAccept}(\log_7, c\_keys', i')) \implies i = i' \quad (33)$$

with public variables  $V_{\text{in}} = \{c\_cats, c\_sats, c\_ems, c\_resumption\_secret, s\_cats1, s\_cats2, s\_sats, s\_ems1, s\_ems2, s\_resumption\_secret1, s\_resumption\_secret2\}$ . The correspondences (30) and (32) prove injective mutual key authentication, and (31) and (33) prove that the accept events are executed at most once for value of the session identifier  $\log_4$  or  $\log_7$ . The session identifier  $\log_7$  contains all messages of the protocol, while  $\log_4$  contains all messages except the last one (client **Finished**). The keys *c\_keys* include all four keys mentioned above, while *s\_keys* does not contain the resumption secret. CryptoVerif also proves that the protocol preserves the secrecy of *s\_sats* with public variables  $V_{\text{in}} \setminus \{c\_sats, s\_sats\}$ , of *c\_cats* with public variables  $V_{\text{in}} \setminus \{c\_cats, s\_cats1, s\_cats2\}$ , of *c\_ems* with public variables  $V_{\text{in}} \setminus \{c\_ems, s\_ems1, s\_ems2\}$ , and of *c\_resumption\_secret* with public variables  $V_{\text{in}} \setminus \{c\_resumption\_secret, s\_resumption\_secret1, s\_resumption\_secret2\}$ . The events are executed and the keys are stored in the corresponding variable only when each participant believes that it talks to a honest peer: when the honest peer is authenticated and not compromised, or when the honest peer is

compromised but the received messages still come from it, or when the peer is not authenticated but the Diffie-Hellman share comes from the honest peer.

**Handshakes with Pre-Shared Key** The handshakes with pre-shared key rely on a previously established pre-shared key to execute a handshake between the client and the server, with or without Diffie-Hellman key exchange. They produce the same keys as the initial handshake, plus an additional client early traffic secret *cets*, computed after the first message of the protocol (**ClientHello**). This traffic secret is used by the record protocol to send messages from the client to the server immediately after the **ClientHello** message, so-called 0-RTT data. These keys are stored in variables with prefix *c\_* client side and *s\_* server side for the handshake without Diffie-Hellman exchange, and *cdhe\_* client side and *sdhe\_* server side for the handshake with Diffie-Hellman exchange. Server-side, *cets* is stored in the variables *s\_cets2* when the **ClientHello** has not been altered by an adversary, *s\_cets3* when it has been altered, and *s\_cets1* when it has been altered and the considered **ClientHello** message has not been received before (and similarly with *dhe* added for the handshake with Diffie-Hellman exchange). CryptoVerif proves the correspondences (30) to (33) with public variables  $V_{\text{psk}}$  containing all variables with prefixes *c\_*, *s\_*, *cdhe\_*, and *sdhe\_*. CryptoVerif also proves that the protocol preserves the secrecy of *s\_sats* with public variables  $V_{\text{psk}} \setminus \{c\_sats, s\_sats\}$ , of *c\_cats* with public variables  $V_{\text{psk}} \setminus \{c\_cats, s\_cats\}$ , of *c\_ems* with public variables  $V_{\text{psk}} \setminus \{c\_ems, s\_ems\}$ , and of *c\_resumption\_secret* with public variables  $V_{\text{psk}} \setminus \{c\_resumption\_secret, s\_resumption\_secret\}$ .

For 0-RTT traffic, CryptoVerif shows the correspondences

$$\text{event}(\text{ServerEarlyTerm1}(\log1, cets)) \implies \text{event}(\text{ClientEarlyAccept1}(\log1, cets, i)) \quad (34)$$

$$\text{event}(\text{ClientEarlyAccept1}(\log1, cets, i)) \wedge \text{event}(\text{ClientEarlyAccept1}(\log1, cets', i')) \implies i = i' \quad (35)$$

with public variables  $V_{\text{psk}}$ , and the secrecy of *c\_cets* with public variables  $V_{\text{psk}} \setminus \{c\_cets, s\_cets2\}$ . These properties deal with the case of **ClientHello** messages that have not been altered by the adversary. The authentication (34) is non-injective because the **ClientHello** message can be replayed.

To deal with the case of altered **ClientHello** messages, CryptoVerif shows the correspondence

$$\text{event}(\text{ServerEarlyTerm2}(\log1, cets)) \wedge \text{event}(\text{ServerEarlyTerm2}(\log1, cets')) \implies cets = cets' \quad (36)$$

with public variables  $V_{\text{psk}}$ , and the secrecy of *s\_cets1* with public variables  $V_{\text{psk}} \setminus \{s\_cets1, s\_cets3\}$ . The correspondence (36) means that, if the server receives twice the same altered **ClientHello** message, then it computes the same early traffic secret *cets*.

CryptoVerif proves similar properties for the handshake with Diffie-Hellman exchange, with suffix DHE added to the events and *dhe* added to the variables.

**Record Protocol** The record protocol is modeled in CryptoVerif as follows:

$$\begin{aligned} \text{Rec} = & c_1(); \text{new } b : \text{bool}; \text{new } ts : \text{key}; \text{let } ts_{\text{upd}} = \text{HKDF\_expand\_upd\_label}(ts) \text{ in } \overline{c_2}(); \\ & (Q_{\text{send}}(b) \mid Q_{\text{recv}}) \end{aligned}$$

It chooses a random bit *b* (*false* = 0 or *true* = 1) and a random traffic secret *ts*. It computes the updated traffic secret *ts<sub>upd</sub>*, and then provides two processes  $Q_{\text{send}}(b)$  and  $Q_{\text{recv}}$ . The process  $Q_{\text{send}}(b)$  receives two clear messages *msg<sub>0</sub>* and *msg<sub>1</sub>*, and a counter *count*. Provided the counter has not been used for sending a previous message and the messages *msg<sub>0</sub>* and *msg<sub>1</sub>* have

the same padded length, it executes the event  $\text{sent}_0(\text{count}, \text{msg}_b)$  and sends the message  $\text{msg}_b$  encrypted using keys derived from the traffic secret  $ts$ . The process  $Q_{\text{recv}}$  receives an encrypted message and a counter  $\text{count}$ . Provided the counter has not been used for receiving a previous message, it decrypts the message using keys derived from the traffic secret  $ts$  and executes event  $\text{received}_0(\text{count}, \text{msg})$  where  $\text{msg}$  is the clear message. Both the emission and reception can be executed several times, and the encryption scheme is authenticated.

CryptoVerif proves the secrecy of  $ts_{\text{upd}}$  with public variable  $b$ , the secrecy of  $b$  with public variable  $ts_{\text{upd}}$ , and the correspondence

$$\text{inj-event}(\text{received}_0(\text{count}, \text{msg})) \implies \text{inj-event}(\text{sent}_0(\text{count}, \text{msg}))$$

with public variables  $b, ts_{\text{upd}}$ . This correspondence shows injective message authentication.

We consider two other variants of the record protocol, used for 0-RTT. In the first variant,  $\text{Rec}_{0\text{-RTT}}$ , the receiver process is replicated once more, so that several sessions may have the same traffic secret, thus the receiver accepts messages with the same counter in different sessions with the same traffic secret. It models that the server may receive several times the same **ClientHello** message, yielding the same traffic secret. In this model, CryptoVerif proves the secrecy of  $ts_{\text{upd}}$  with public variable  $b$ , the secrecy of  $b$  with public variable  $ts_{\text{upd}}$ , and the correspondence

$$\text{event}(\text{received}_0(\text{count}, \text{msg})) \implies \text{event}(\text{sent}_0(\text{count}, \text{msg}))$$

with public variables  $b, ts_{\text{upd}}$ .

In the second variant,  $\text{Rec}_{0\text{-RTT}, \text{Bad}}$ , the sender process is additionally removed. This model corresponds to the situation in which the **ClientHello** message is altered, and thus the server obtains a traffic secret that is not used by any client. In this model, CryptoVerif proves the secrecy of  $ts_{\text{upd}}$  with no public variable and the correspondence  $\text{event}(\text{received}_0(\text{count}, \text{msg})) \implies \text{false}$  with public variable  $ts_{\text{upd}}$ , that is,  $\text{event}(\text{received}_0(\text{count}, \text{msg}))$  can be executed only with negligible probability.

## B.2 The Composition

Let us now apply our composition theorems to compose the three parts of TLS 1.3 into the whole protocol.

**Record Protocol with Key Updates** We compose the record protocol with itself, to deal with key updates.

As a first step, let us establish properties of the record protocol without composition. For  $s \in \{0, 1\}$ , let  $\text{Rec}_s$  be the protocol obtained by always choosing  $b = s$  in the record protocol  $\text{Rec}$ . Let  $\text{Rec}_s^\circ$  be obtained by removing events in  $\text{Rec}_s$ . Let  $\text{Rec}_\star$  be the protocol in which the choice of  $b$  is removed and the sender receives a single clear message and sends it encrypted. We let  $\text{corr}_j$  be the correspondence

$$\text{inj-event}(\text{received}_j(\text{count}, \text{msg})) \implies \text{inj-event}(\text{sent}_j(\text{count}, \text{msg}))$$

We prove that  $\text{Rec}_0^\circ \approx^V \text{Rec}_1^\circ$  and for all  $s \in \{0, 1, \star\}$ ,  $\text{Rec}_s$  preserves the secrecy of  $ts_{\text{upd}}$  and satisfies the correspondence  $\text{corr}_0$  with public variables  $V$ , where  $V = \{ts_{\text{upd}}\}$ .

*Proof.* By Lemma 12, since  $\text{Rec}$  preserves the secrecy of  $b$  with public variables  $V$ , we have  $\text{Rec}_0^\circ \approx^V \text{Rec}_1^\circ$ .

Moreover, since  $\text{Rec}$  preserves the secrecy of  $ts_{\text{upd}}$  and satisfies the correspondence  $\text{corr}_0$  with public variables  $V$ , so do  $\text{Rec}_0$  and  $\text{Rec}_1$  by Lemma 1, because for  $s \in \{0, 1\}$ ,  $\text{Rec}_s \approx^V C_s[\text{Rec}]$

and the evaluation context  $C_s$  receives two messages  $msg_0$  and  $msg_1$  and forwards two copies of  $msg_s$  to  $Rec$ , so that  $Rec$  always sends  $msg_s$  encrypted independently of the value of  $b$ .

Furthermore, since  $Rec$  preserves the secrecy of  $ts_{upd}$  and satisfies the correspondence  $corr_0$  with public variables  $V$ ,  $Rec_\star$  satisfies the same properties by Lemma 1, because  $Rec_\star \approx^V C[Rec]$ , where the evaluation context  $C$  receives a single message  $msg$  and forwards two copies of  $msg$  to  $Rec$ , so that  $Rec$  always sends  $msg_0 = msg_1 = msg$  encrypted independently of the value of  $b$ .  $\square$

Let us now define the record protocol with  $m$  key updates as follows:

$$Rec_s^m = c_1(); \mathbf{new} \ ts : key; \overline{c_2} \langle \rangle; \left( \prod_{j=0}^m Q_{send,s}^j \mid \prod_{j=0}^m Q_{recv}^j \right)$$

where  $s$  is the side (0, 1, or  $\star$ ),  $Q_{send,s}^j$  is a process that receives messages on channel  $c_3^j$  and sends them encrypted using the  $j$ -th updated traffic secret on channel  $c_4^j$ , and  $Q_{recv}^j$  is a process that receives messages on channel  $c_5^j$  and decrypts them using the  $j$ -th updated traffic secret. The process  $Q_{send,s}^j$  uses table  $table\_count\_send^j$  and event  $\mathbf{sent}_j$ ;  $Q_{recv}^j$  uses table  $table\_count\_recv^j$  and event  $\mathbf{received}_j$ . When  $s$  is 0 or 1,  $Q_{send,s}^j$  receives two clear messages  $msg_0$  and  $msg_1$  and sends  $msg_s$  encrypted. When  $s = \star$ ,  $Q_{send,s}^j$  receives a single message and sends it encrypted. Let  $Rec_s^{m,\circ}$  be obtained by removing all events in  $Rec_s^m$ .

The record protocol can be written:

$$Rec_s = c_1(); \mathbf{new} \ ts : key; \mathbf{let} \ ts_{upd} = \mathbf{HKDF\_expand\_upd\_label}(ts) \mathbf{in} \ \overline{c_2} \langle \rangle; \\ (Q_{send,s}^0 \mid Q_{recv}^0)$$

We prove that  $Rec_0^{m,\circ} \approx Rec_1^{m,\circ}$  and for all  $s \in \{0, 1, \star\}$ ,  $Rec_s^m$  satisfies the correspondences  $corr_j$  with no public variable for  $j \leq m$ .

*Proof.* The proof proceeds by induction on  $m$ .

*Case  $m = 0$ :* Since  $Rec_0^\circ \approx^V Rec_1^\circ$  and for all  $s \in \{0, 1, \star\}$ ,  $Rec_s$  satisfies the correspondence  $corr_0$  with public variables  $ts_{upd}$ , we have  $Rec_0^{0,\circ} \approx Rec_1^{0,\circ}$  and for all  $s \in \{0, 1, \star\}$ ,  $Rec_s^0$  satisfies the correspondence  $corr_0$  with no public variable, by removing the assignment to  $ts_{upd}$ .

*Case  $m+1$ :* By induction hypothesis,  $Rec_0^{m,\circ} \approx Rec_1^{m,\circ}$  and for all  $s \in \{0, 1, \star\}$ ,  $Rec_s^m$  satisfies the correspondences  $corr_j$  with no public variable for  $j \leq m$ . Let  $Rec_s'^m$  be obtained from  $Rec_s^m$  by renaming  $ts$  to  $ts_{upd}$  and other variables to fresh variables, the events  $\mathbf{received}_j$  and  $\mathbf{sent}_j$  to  $\mathbf{received}_{j+1}$  and  $\mathbf{sent}_{j+1}$  respectively for  $j \leq m$ , the channels  $c_1, c_2, c_3^j, c_4^j, c_5^j$  to  $c'_1, c'_2, c_3'^{j+1}, c_4'^{j+1}, c_5'^{j+1}$  respectively for  $j \leq m$ , and the tables  $table\_count\_send^j$  and  $table\_count\_recv^j$  to  $table\_count\_send^{j+1}$  and  $table\_count\_recv^{j+1}$  respectively for  $j \leq m$ . The processes  $Q_{send,s}^j$  and  $Q_{recv}^j$  are defined in a similar way.

We compose  $Rec_s$  with  $Rec_s'^m$  by Theorem 1. By the previous renaming,  $ts_{upd}$  is the only common variable between the processes  $Rec_s$  and  $Rec_s'^m$ , and these processes have no common channel, no common event, and no common table. We obtain the composed system

$$S_{composed,s,s'} = c_1(); \mathbf{new} \ ts : key; \mathbf{let} \ ts_{upd} = \mathbf{HKDF\_expand\_upd\_label}(ts) \mathbf{in} \ \overline{c_2''} \langle \rangle; \\ (Q_{send,s}^0 \mid Q_{recv}^0 \mid \prod_{j=0}^m Q_{send,s'}^j \mid \prod_{j=0}^m Q_{recv}^j)$$

for some fresh channel  $c_2''$ , so that we have

$$S_{composed,s,s'} \{c_2/c_2''\} \approx Rec_s^{m+1}$$

Let  $S_{composed,s,s'}^\circ$  be obtained from  $S_{composed,s,s'}$  by removing events  $\text{received}_0$  and  $\text{sent}_0$ , and  $S_{composed,s,s'}^{\circ\circ}$  be obtained from  $S_{composed,s,s'}$  by removing all events. Since  $\text{Rec}_s$  preserves the secrecy of  $ts_{upd}$ , Theorem 1 shows that  $S_{composed,s,s'}^\circ \approx C'_s[\text{Rec}'_s{}^m]$  for some evaluation context  $C'_s$  acceptable for  $\text{Rec}'_s{}^m$  without public variables and containing no event. Moreover, it also shows that  $S_{composed,s,s'} \approx C''_s[\text{Rec}_s]$  for some evaluation context  $C''_s$  acceptable for  $\text{Rec}_s$  with public variable  $ts_{upd}$  and containing the events  $\text{received}_j$  and  $\text{sent}_j$  for  $1 \leq j \leq m+1$ . Indeed,  $C'_s$  does not depend on the side  $s'$  of  $\text{Rec}'_s{}^m$  and  $C''_s$  does not depend on the side  $s$  of  $\text{Rec}_s$ .

Since  $S_{composed,s,s} \approx C''_s[\text{Rec}_s]$  and  $\text{Rec}_s$  satisfies the correspondence  $\text{corr}_0$  with public variables  $ts_{upd}$ , then by Lemma 1,  $S_{composed,s,s}$  satisfies the correspondence  $\text{corr}_0$  with no public variable, and so does  $\text{Rec}_s^{m+1}$ . Since

$$S_{composed,s,s}^\circ \approx C'_s[\text{Rec}'_s{}^m]$$

and  $\text{Rec}'_s{}^m$  satisfies the correspondences  $\text{corr}_j$  with no public variable for  $1 \leq j \leq m+1$ , then by Lemma 1 so do  $S_{composed,s,s}^\circ$ ,  $S_{composed,s,s}$ , and  $\text{Rec}_s^{m+1}$ . Hence  $\text{Rec}_s^{m+1}$  satisfies the correspondences  $\text{corr}_j$  with no public variable for  $j \leq m+1$ .

Finally, let  $C_0^{\circ\circ}$  be obtained from  $C_0''$  by removing the events  $\text{received}_j$  and  $\text{sent}_j$  for  $1 \leq j \leq m+1$ . Using Lemma 11,

$$\begin{aligned} S_{composed,0,0}^{\circ\circ} &\approx C_0^{\circ\circ}[\text{Rec}_0^\circ] \approx C_0^{\circ\circ}[\text{Rec}_1^\circ] \approx S_{composed,1,0}^{\circ\circ} \\ &\approx C_1'[\text{Rec}_0^{\prime m,\circ}] \approx C_1'[\text{Rec}_1^{\prime m,\circ}] \approx S_{composed,1,1}^{\circ\circ} \end{aligned} \quad (37)$$

so  $\text{Rec}_0^{m+1,\circ} \approx \text{Rec}_1^{m+1,\circ}$  by renaming  $c_2''$  to  $c_2$ .  $\square$

We can apply a similar reasoning to the two variants of the record protocol for 0-RTT data.

- For  $s \in \{0, 1\}$ , let  $\text{Rec}_{0\text{-RTT}_s}^m$  be the process obtained by always choosing  $b = s$  in  $\text{Rec}_{0\text{-RTT}}$  and composing it with itself  $m$  times. Let  $\text{Rec}_{0\text{-RTT}_\star}^m$  be the process  $\text{Rec}_{0\text{-RTT}}$  in which the choice of  $b$  is removed and the sender receives a single clear message and sends it encrypted, composed with itself  $m$  times. Let  $\text{Rec}_{0\text{-RTT}_s}^{m,\circ}$  be obtained from  $\text{Rec}_{0\text{-RTT}_s}^m$  by removing all events. We prove that  $\text{Rec}_{0\text{-RTT}_0}^{m,\circ} \approx \text{Rec}_{0\text{-RTT}_1}^{m,\circ}$  and that, for all  $s \in \{0, 1, \star\}$ ,  $\text{Rec}_{0\text{-RTT}_s}^m$  satisfies the correspondences

$$\text{event}(\text{received}_j(\text{count}, \text{msg})) \implies \text{event}(\text{sent}_j(\text{count}, \text{msg}))$$

with no public variable for all  $j \leq m$ .

- We also prove that the process  $\text{Rec}_{0\text{-RTT,Bad}}^m$ , obtained by composing  $\text{Rec}_{0\text{-RTT,Bad}}$  with itself  $m$  times, satisfies the correspondences

$$\text{event}(\text{received}_j(\text{count}, \text{msg})) \implies \text{false}$$

with no public variable for all  $j \leq m$ . (This variant contains no sender process, so it does not use  $b$ .)

By Lemmas 5 and 6, we infer that replicated versions of the record protocol satisfy similar properties. Let us define  $\text{Repl}_C(Q) = \text{AddReplSid}(i_C \leq n_C, c'_1, \text{bitstring}, Q)$  and  $\text{Repl}_S(Q) = \text{AddReplSid}(i_S \leq n_S, c'_1, \text{bitstring}, Q)$ .

- $\text{Repl}_C(\text{Rec}_0^{m,\circ}) \approx \text{Repl}_C(\text{Rec}_1^{m,\circ})$  and for all  $s \in \{0, 1, \star\}$ ,  $\text{Repl}_C(\text{Rec}_s^m)$  satisfies the correspondences  $\text{inj-event}(\text{received}_j(x, \text{count}, \text{msg})) \implies \text{inj-event}(\text{sent}_j(x, \text{count}, \text{msg}))$  with no public variable for all  $j \leq m$ .



- $Repl_S(Rec_s^m)$  satisfies the same properties.
- $Repl_C(Rec_{0-RTT_0}^{m,\circ}) \approx Repl_C(Rec_{0-RTT_1}^{m,\circ})$  and for all  $s \in \{0, 1, \star\}$ ,  $Repl_C(Rec_{0-RTT_s}^m)$  satisfies the correspondences  $\mathbf{event}(\mathbf{received}_j(x, count, msg)) \implies \mathbf{event}(\mathbf{sent}_j(x, count, msg))$  with no public variable for all  $j \leq m$ .
- $Repl_S(Rec_{0-RTT, \text{Bad}}^m)$  satisfies the correspondences  $\mathbf{event}(\mathbf{received}_j(x, count, msg)) \implies \mathbf{false}$  with no public variable for all  $j \leq m$ .

**Handshakes with Pre-Shared Key** We compose the handshakes with pre-shared key with the record protocol:

1. with secret key  $c\_cats$  using Theorem 2 and process  $Repl_C(Rec_s^m)$ , for 1-RTT client-to-server messages. (Event  $e_A$  is **ClientAccept** and event  $e_B$  is **ServerTerm**. In [10, Figure 9], event **ClientAccept** occurs at line 3: and event **ServerTerm** occurs at line 7:.)
2. with secret key  $s\_sats$  using Theorem 2 and process  $Repl_S(Rec_s^m)$ , for 0.5-RTT and 1-RTT server-to-client messages. (Event  $e_A$  is **ServerAccept** and event  $e_B$  is **ClientTerm**. In [10, Figure 9], event **ServerAccept** occurs at line 6: and event **ClientTerm** occurs at line 2:.)
3. with secret key  $c\_cets$  using Theorem 3 and the variant  $Repl_C(Rec_{0-RTT_s}^m)$  of the record protocol with an additional replication in the receiver process, for 0-RTT messages when the **ClientHello** message has not been altered. (Event  $e_A$  is **ClientEarlyAccept1** and event  $e_B$  is **ServerEarlyTerm1**. In [10, Figure 9], event **ClientEarlyAccept1** occurs at line 1: and event **ServerEarlyTerm1** occurs at line 4:.)
4. with secret key  $s\_cets3$  using Theorem 5 and the variant  $Repl_S(Rec_{0-RTT, \text{Bad}}^m)$  of the record protocol without sender process, for 0-RTT when the **ClientHello** message has been altered. (Event  $e$  is **ServerEarlyTerm2**, which occurs at line 5: in [10, Figure 9].)

We perform similar compositions (numbered 1' to 4') in the handshake with pre-shared key and Diffie-Hellman key agreement.

Before applying the theorems, we permute assignments and events as needed so that the process has the form required in the theorem. We also notice that the events contain more keys in the process than in the composition theorems: the handshake process contains events  $e_A((\widetilde{msg}_A), (\widetilde{k}_A), \widetilde{i})$  and  $e_B((\widetilde{msg}_B), (\widetilde{k}_B))$  while in the composition theorems, the process  $Q'_1$  contains  $e_A((\widetilde{msg}_A), k_A, \widetilde{i})$  and  $e_B((\widetilde{msg}_B), k_B)$ , where  $\widetilde{k}_A$  contains  $k_A$  and  $\widetilde{k}_B$  contains  $k_B$  at the same position. (The function  $\text{sid}$  just returns the tuple of messages.) As a result, the correspondences (2) and (7) that we prove are stronger than those in the composition theorems: they mean that for each execution of event  $e_B((\widetilde{msg}), (\widetilde{k}))$ , there is an execution of event  $e_A((\widetilde{msg}), (\widetilde{k}), \widetilde{i})$  for some  $\widetilde{i}$ , with the same tuples of keys  $\widetilde{k} = \widetilde{k}_A = \widetilde{k}_B$  while the composition theorems just require the equality for one key  $k = k_A = k_B$ . The meaning of the correspondence (3) is unchanged, since in this correspondence, the keys are distinct variables so their value does not matter.

In the composition, we index the events that come from the record protocol with an index  $c$  from 1 to 4 and from 1' to 4' corresponding to the number of the composition that adds this instance of the record protocol, so we use events  $\text{sent}_{c,j}$  and  $\text{received}_{c,j}$ . We index the channels, tables, and variables from the record protocol in a similar way, so that they are distinct in the various compositions.

For the first composition, we compose the handshakes with pre-shared key  $Q_{\text{PSKH}}$  with the record protocol  $Rec_s^{1,m}$ , obtained from  $Repl_C(Rec_s^m)$  by this renaming, with secret key

$c\_cats$  using Theorem 2, as announced above. We obtain a composed protocol  $S_{composed,s}^{1,m}$ . Let  $S_{composed,s}^{1,m,\circ}$  be obtained from  $S_{composed,s}^{1,m}$  by removing the events of the handshake protocol, and  $S_{composed,s}^{1,m,\circ\circ}$  be obtained from  $S_{composed,s}^{1,m}$  by removing all events. By Theorem 2, we have  $S_{composed,s}^{1,m,\circ} \xrightarrow{f_1^{V_{psk}^1, \emptyset}} Rec_s^{1,m}$  and  $S_{composed,s}^{1,m} \approx^{V_{psk}^1} C_s^1[Q_{PSKH}]$  for some evaluation context  $C_s^1$  acceptable for  $Q_{PSKH}$  with public variables  $\{c\_cats, s\_cats\}$ , where  $V_{psk}^1 = V_{psk} \setminus \{c\_cats, s\_cats\}$ . By Lemma 11, we have  $S_{composed,s}^{1,m,\circ\circ} \xrightarrow{f_1^{V_{psk}^1, \emptyset}} Rec_s^{1,m,\circ}$ , where the process  $Rec_s^{1,m,\circ}$  is obtained by deleting events in  $Rec_s^{1,m}$ . Since  $Repl_C(Rec_0^{m,\circ}) \approx Repl_C(Rec_1^{m,\circ})$ , that is,  $Rec_0^{1,m,\circ} \approx Rec_1^{1,m,\circ}$ , we have  $S_{composed,0}^{1,m,\circ\circ} \approx^{V_{psk}^1} S_{composed,1}^{1,m,\circ\circ}$  by Lemma 2. By Lemma 3 and  $S_{composed,s}^{1,m,\circ} \xrightarrow{f_1^{V_{psk}^1, \emptyset}} Rec_s^{1,m}$ , the processes  $S_{composed,s}^{1,m,\circ}$  and  $S_{composed,s}^{1,m}$  inherit the correspondence properties of the record protocol. By Lemma 1 and  $S_{composed,s}^{1,m} \approx^{V_{psk}^1} C_s[Q_{PSKH}]$ , the process  $S_{composed,s}^{1,m}$  inherits correspondence and secrecy properties of the handshake with pre-shared keys (but the variable  $c\_cats$  on which we compose and the corresponding variable  $s\_cats$  in the peer are removed from the public variables  $V_{psk}^1$  and from the secrets).

We perform the second composition in a similar way. We compose  $S_{composed,s}^{1,m}$  with the record protocol  $Rec_s^{2,m}$ , obtained by renaming  $Repl_S(Rec_s^m)$ , with secret key  $s\_sats$  using Theorem 2, as announced above. We obtain a composed protocol  $S_{composed,s,s'}^{2,m}$ . Let  $S_{composed,s,s'}^{2,m,\circ}$  be obtained from  $S_{composed,s,s'}^{2,m}$  by removing the events of  $S_{composed,s}^{1,m}$ , and  $S_{composed,s,s'}^{2,m,\circ\circ}$  be obtained from  $S_{composed,s,s'}^{2,m}$  by removing all events. By Theorem 2, we have  $S_{composed,s,s'}^{2,m,\circ} \xrightarrow{f_s^{V_{psk}^2, \emptyset}} Rec_{s'}^{2,m}$  and  $S_{composed,s,s'}^{2,m} \approx^{V_{psk}^2} C_{s'}^2[S_{composed,s}^{1,m}]$  for some evaluation context  $C_{s'}^2$  acceptable for  $S_{composed,s}^{1,m}$  with public variables  $\{s\_sats, c\_sats\}$ , where  $V_{psk}^2 = V_{psk}^1 \setminus \{s\_sats, c\_sats\}$ . By Lemma 11, we have  $S_{composed,s,s'}^{2,m,\circ\circ} \xrightarrow{f_s^{V_{psk}^2, \emptyset}} Rec_{s'}^{2,m,\circ}$ , where the process  $Rec_{s'}^{2,m,\circ}$  is obtained by deleting events in  $Rec_{s'}^{2,m}$ . Since  $Repl_S(Rec_0^{m,\circ}) \approx Repl_S(Rec_1^{m,\circ})$ , that is,  $Rec_0^{2,m,\circ} \approx Rec_1^{2,m,\circ}$ , we have

$$S_{composed,0,0}^{2,m,\circ\circ} \approx^{V_{psk}^2} S_{composed,0,1}^{2,m,\circ\circ}$$

by Lemma 2. Moreover, using Lemma 11,

$$S_{composed,0,1}^{2,m,\circ\circ} \approx^{V_{psk}^2} C_1^{2,\circ}[S_{composed,0}^{1,m,\circ\circ}] \approx^{V_{psk}^2} C_1^{2,\circ}[S_{composed,1}^{1,m,\circ\circ}] \approx^{V_{psk}^2} S_{composed,1,1}^{2,m,\circ\circ}$$

so

$$S_{composed,0,0}^{2,m,\circ\circ} \approx^{V_{psk}^2} S_{composed,1,1}^{2,m,\circ\circ}.$$

By Lemma 3 and  $S_{composed,s,s}^{2,m,\circ} \xrightarrow{f_s^{V_{psk}^2, \emptyset}} Rec_s^{2,m}$ , the processes  $S_{composed,s,s}^{1,m,\circ}$  and  $S_{composed,s,s}^{1,m}$  inherit the correspondence properties of the record protocol. By Lemma 1 and  $S_{composed,s,s}^{2,m} \approx^{V_{psk}^2} C_s^2[S_{composed,s}^{1,m}]$ , the process  $S_{composed,s,s}^{2,m}$  inherits correspondence and secrecy properties of the handshake with pre-shared keys (but the variable  $s\_sats$  on which we compose and the corresponding variable  $c\_sats$  in the peer are removed from the public variables  $V_{psk}^2$  and from the secrets).

We perform the other compositions in the same way. We obtain processes  $Q_{PSKH,s}^m$  that run one handshake with pre-shared key and the record protocol with at most  $m$  key updates and side  $s$ . Let  $Q_{PSKH,s}^{m,\circ}$  be obtained from  $Q_{PSKH,s}^m$  by removing all events. The composition theorems show that  $Q_{PSKH,0}^{m,\circ} \approx^V Q_{PSKH,1}^{m,\circ}$  where  $V = \{c\_ems, s\_ems, cdhe\_ems, sdhe\_ems, c\_resumption\_secret, s\_resumption\_secret, cdhe\_resumption\_secret, sdhe\_resumption\_secret\}$ , and that  $Q_{PSKH,s}^m$  satisfies correspondences and secrecy properties

that come from the handshake (but the variables  $c\_cats$ ,  $s\_sats$ ,  $c\_cets$ ,  $s\_cets3$ ,  $cdhe\_cats$ ,  $sdhe\_sats$ ,  $cdhe\_cets$ , and  $sdhe\_cets3$  on which we compose and the corresponding variables in the peer are removed from the public variables  $V$  and from the secrets) and from the record protocol (with additional index  $c$  in the correspondences). That is,  $Q_{PSKH,s}^m$  satisfies the correspondences (30) to (36), the correspondences (30) to (36) with suffix DHE in the events, and

$$\begin{aligned} \text{inj-event}(\text{received}_{c,j}(x, \text{count}, \text{msg})) &\implies \text{inj-event}(\text{sent}_{c,j}(x, \text{count}, \text{msg})) \\ &\text{for } c \in \{1, 2, 1', 2'\} \text{ and } j \leq m \\ \text{event}(\text{received}_{c,j}(x, \text{count}, \text{msg})) &\implies \text{event}(\text{sent}_{c,j}(x, \text{count}, \text{msg})) \\ &\text{for } c \in \{3, 3'\} \text{ and } j \leq m \\ \text{event}(\text{received}_{c,j}(x, \text{count}, \text{msg})) &\implies \text{false} \\ &\text{for } c \in \{4, 4'\} \text{ and } j \leq m \end{aligned}$$

with public variables  $V$  and preserves the secrecy of

$$\begin{aligned} &c\_ems \text{ with public variables } V \setminus \{c\_ems, s\_ems\}, \\ &cdhe\_ems \text{ with public variables } V \setminus \{cdhe\_ems, sdhe\_ems\}, \\ &c\_resumption\_secret \text{ with public variables} \\ &\quad V \setminus \{c\_resumption\_secret, s\_resumption\_secret\}, \\ &cdhe\_resumption\_secret \text{ with public variables} \\ &\quad V \setminus \{cdhe\_resumption\_secret, sdhe\_resumption\_secret\} \end{aligned}$$

Let us define processes  $Q_{PSKH,s}^{l,m}$  that run at most  $l+1$  successive handshakes with pre-shared key and the record protocol with at most  $m$  key updates and side  $s$ . In these processes, we add an index  $k$  to all events:  $k$  is a bitstring of length at most  $l$ , where the length of  $k$  is the number of handshakes with pre-shared key made before the current one and the  $i$ -th bit of  $k$  is 1 if and only if the  $i$ -th handshake used a Diffie-Hellman exchange. The index  $k$  is also added to the variables  $c\_ems$ ,  $s\_ems$ ,  $cdhe\_ems$ ,  $sdhe\_ems$ . We also add arguments  $\tilde{x}_k$  to events, where  $\tilde{x}_k$  is a sequence of variables that will contain the messages sent and received by the handshakes above the current one. The sequence  $\tilde{x}_k$  contains as many variables as the length of  $k$ . The processes  $Q_{PSKH,s}^{l,m}$  start with a context  $C_h$  that defines a random oracle:  $Q_{PSKH,s}^{l,m} = C_h[Q_{PSKH\text{-no-ROM},s}^{l,m}]$ . Hence, the corresponding replicated process constructed by Lemmas 5 and 6 is  $Q_{PSKH\text{-Repl},s}^{l,m} = C'_h[\text{AddReplSid}(i_C \leq n_C, c', \text{bitstring}, Q_{PSKH\text{-no-ROM},s}^{l,m})]$ . Let  $Q_{PSKH,s}^{l,m,\circ}$  be obtained from  $Q_{PSKH,s}^{l,m}$  by removing all events, and  $Q_{PSKH\text{-Repl},s}^{l,m,\circ}$  be obtained from  $Q_{PSKH\text{-Repl},s}^{l,m}$  by removing all events.

We have  $Q_{PSKH,s}^{0,m} = Q_{PSKH,s}^m$ . Using a proof by induction as for the record protocol above, from properties of  $Q_{PSKH,s}^{l,m}$ , we infer properties of  $Q_{PSKH\text{-Repl},s}^{l,m}$  by Lemmas 5 and 6 and we compose  $Q_{PSKH,s}^m$  with  $Q_{PSKH\text{-Repl},s}^{l,m}$ , with secret  $c\_resumption\_secret$  using Theorem 2. (Event  $e_A$  is `ClientAccept` and event  $e_B$  is `ServerTerm`.) We perform a similar composition with secret  $cdhe\_resumption\_secret$ . Then, we obtain properties of  $Q_{PSKH,s}^{l+1,m}$ .

We obtain that  $Q_{PSKH,0}^{l,m,\circ} \approx^{V_l} Q_{PSKH,1}^{l,m,\circ}$  and that  $Q_{PSKH,s}^{l,m}$  satisfies the same correspondences as  $Q_{PSKH,s}^m$ , with additional index  $k$  of length at most  $l$  and additional arguments  $\tilde{x}_k$  in the events and with public variables  $V_l$ , and preserves the secrecy of  $c\_ems_k$  with public variables  $V_l \setminus \{c\_ems_k, s\_ems_k\}$  and of  $cdhe\_ems_k$  with public variables  $V_l \setminus \{cdhe\_ems_k, sdhe\_ems_k\}$ , where  $V_l = \bigcup_{k \text{ of length at most } l} \{c\_ems_k, s\_ems_k, cdhe\_ems_k, sdhe\_ems_k\}$ .

We also obtain that  $Q_{\text{PSKH-Repl},0}^{l,m,\circ} \approx_{V_l} Q_{\text{PSKH-Repl},1}^{l,m,\circ}$  and that  $Q_{\text{PSKH-Repl},s}^{l,m}$  satisfies the same correspondences as  $Q_{\text{PSKH},s}^m$ , with additional index  $k$  of length at most  $l$  and additional arguments  $x, \tilde{x}_k$  in the events and with public variables  $V_l$ , and preserves the secrecy of  $c\_ems_k$  with public variables  $V_l \setminus \{c\_ems_k, s\_ems_k\}$  and of  $cdhe\_ems_k$  with public variables  $V_l \setminus \{cdhe\_ems_k, sdhe\_ems_k\}$ .

**Full Protocol** We compose the initial handshake with the record protocol:

1. with secret key  $c\_cats$  using Theorem 4 and process  $\text{AddReplSid}(i_C \leq n_C, c', \text{bitstring}, \text{Rec}_s^m)$ , for 1-RTT client-to-server messages. (Event  $e_A$  is **ClientAccept** and event  $e_B$  is **ServerTerm**. In [10, Figure 8], event **ClientAccept** occurs at line 2: and event **ServerTerm** occurs at line 6: and in the process  $Q_{\text{ServerAfter}0.5RTT2}$ . We need the variant with several holes of Theorem 2, shown in Appendix A.10, because event **ServerTerm** occurs twice.)
2. with secret key  $c\_sats$  using Theorem 2 and process  $\text{AddReplSid}(i_S \leq n_S, c', \text{bitstring}, \text{Rec}_s^m)$ , for 0.5-RTT and 1-RTT server-to-client messages. (Event  $e_A$  is **ServerAccept** and event  $e_B$  is **ClientTerm**. In [10, Figure 8], event **ServerAccept** occurs at line 4: and event **ClientTerm** occurs at line 1:.)

In the composition, we index the events that come from the record protocol with an index  $c$  (1 or 2) corresponding to the number of the composition that adds this instance of the record protocol, so we use events  $\text{sent}_{c,j}$  and  $\text{received}_{c,j}$ .

We also compose the initial handshake with the process  $Q_{\text{PSKH-Repl},s}^{l,m}$  that runs handshakes with pre-shared key, with secret key  $c\_resumption\_secret$  using Theorem 4. (Event  $e_A$  is **ClientAccept** and event  $e_B$  is **ServerTerm**. Again, we need the variant with several holes of Theorem 2, because event **ServerTerm** occurs twice.)

Finally, we compose the obtained process with a process that runs the rest of the TLS protocol (record protocol and/or handshakes with pre-shared key) without any event or security claim, when the honest client talks to a dishonest server or conversely. In these cases, the model leaks the session keys, so we just put the process in a context that receives these keys and runs the rest of TLS, and perform the composition by eliminating communications (Appendix A.2). In [10, Figure 8], this situation occurs at lines 3:, 5:, 7:, and at a line similar to 7: in the process  $Q_{\text{ServerAfter}0.5RTT2}$ .

We obtain processes  $Q_{\text{TLS},s}^{l,m}$  that run the initial handshake followed by at most  $l$  successive handshakes with pre-shared key and the record protocol with at most  $m$  key updates and side  $s$ . Let  $Q_{\text{TLS},s}^{l,m,\circ}$  be obtained from  $Q_{\text{TLS},s}^{l,m}$  by removing all events.

These processes satisfy the following properties:  $Q_{\text{TLS},0}^{l,m,\circ} \approx_{V_l} Q_{\text{TLS},1}^{l,m,\circ}$  and  $Q_{\text{TLS},s}^{l,m}$  satisfies the correspondences (30) to (33) (inherited from the initial handshake), the correspondences (30) to (36) with additional index  $k$  and additional arguments  $x, \tilde{x}_k$  in the events, the correspondences (30) to (36) with suffix DHE, additional index  $k$ , and additional arguments  $x, \tilde{x}_k$  in the events,

$$\text{inj-event}(\text{received}_{c,j}(x', \text{count}, \text{msg})) \implies \text{inj-event}(\text{sent}_{c,j}(x', \text{count}, \text{msg})) \quad (38)$$

for  $c \in \{1, 2, 1', 2'\}$  and  $j \leq m$

$$\text{inj-event}(\text{received}_{k,c,j}(x', \tilde{x}_k, x, \text{count}, \text{msg})) \implies \text{inj-event}(\text{sent}_{k,c,j}(x', \tilde{x}_k, x, \text{count}, \text{msg})) \quad (39)$$

for  $c \in \{1, 2, 1', 2'\}$  and  $j \leq m$

$$\text{event}(\text{received}_{k,c,j}(x', \tilde{x}_k, x, \text{count}, \text{msg})) \implies \text{event}(\text{sent}_{k,c,j}(x', \tilde{x}_k, x, \text{count}, \text{msg})) \quad (40)$$

for  $c \in \{3, 3'\}$  and  $j \leq m$

$$\begin{aligned} \text{event}(\text{received}_{k,c,j}(x', \tilde{x}_k, x, \text{count}, \text{msg})) &\implies \text{false} \\ \text{for } c \in \{4, 4'\} \text{ and } j \leq m \end{aligned} \quad (41)$$

all with public variables  $V_l'$  and preserves the secrecy of

$$\begin{aligned} &c\_ems \text{ with public variables } V_l' \setminus \{c\_ems, s\_ems1, s\_ems2\}, \\ &c\_ems_k \text{ with public variables } V_l' \setminus \{c\_ems_k, s\_ems_k\}, \\ &cdhe\_ems_k \text{ with public variables } V_l' \setminus \{cdhe\_ems_k, sdhe\_ems_k\} \end{aligned}$$

where

$$V_l' = \{c\_ems, s\_ems1, s\_ems2\} \cup \bigcup_{k \text{ of length at most } l} \{c\_ems_k, s\_ems_k, cdhe\_ems_k, sdhe\_ems_k\}.$$

(The events and variables with additional index  $k$  are considered different from the events and variables without that index, even when  $k$  is empty. Those without index  $k$  come from the initial handshake. Those with index  $k$  come from the handshake with pre-shared key.)

The equivalence  $Q_{\text{TLS},0}^{l,m,\circ} \approx^{V_l'} Q_{\text{TLS},1}^{l,m,\circ}$  proves message secrecy: an adversary cannot distinguish whether the protocol encrypted the first or the second set of plaintexts. The correspondences (38) and (39) prove injective message authentication for 0.5-RTT and 1-RTT data, while the correspondences (40) prove non-injective message authentication for 0-RTT data: if a honest receiver is in a honest session (that is, it talks to a honest sender) and receives a message, then this message was sent by the honest sender talking to the honest receiver. Furthermore, the message has the same associated counter *count* on both sides. Other correspondences show (injective or non-injective) key authentication, and unique accept for the composed protocol. We also obtain the secrecy of the exporter master secrets computed by the various handshakes: the adversary cannot distinguish them from independent random values.

**Remark** The correspondence (31) with additional index  $k$  and additional arguments  $x, \tilde{x}_k$  in the events is

$$\begin{aligned} &\text{event}(\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys, i)) \wedge \\ &\text{event}(\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys', i')) \implies i = i' \end{aligned} \quad (42)$$

It shows that, when two executions of  $\text{ServerAccept}_k$  with the same log  $(x, \tilde{x}_k, \log_4)$  from the beginning of the protocol always have the same final replication index  $i$ . However, the intuitive extension of (31) to the composed system is rather that  $\text{ServerAccept}_k$  is executed at most once with the same log, that is, two executions of  $\text{ServerAccept}_k$  with the same log have *all* their replication indices equal:

$$\begin{aligned} &\text{event}(\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys, i_0, \tilde{i}_k, i)) \wedge \\ &\text{event}(\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys', i'_0, \tilde{i}'_k, i')) \implies (i_0, \tilde{i}_k, i) = (i'_0, \tilde{i}'_k, i') \end{aligned} \quad (43)$$

where  $\tilde{i}_k$  contains as many replication indices as the length of  $k$ .

In general,  $\text{AddReplSid}$  can add only the session identifier to the events, and not the replication indices, because the replication indices are renumbered during the composition. (With the notations of Theorem 2, the indices of  $Q_{2B}$  are renumbered.) That would break some correspondences. In the particular case of (31), we could actually add the indices because either the indices of both occurrences of  $\text{ServerAccept}_k$  are not renumbered or they are renumbered in the same way, so the correspondence is preserved.

Formally, we do not need (43), because in all our compositions, we use as key exchange protocol (system  $S_1$  in Theorem 2) a protocol that performs a single handshake, so (31) itself is enough. We may still want to prove (43). We can show it by combining several correspondences: in the composed protocol, the event  $\text{ServerAccept}_k$  is always preceded by the events  $\text{ServerAccept}$  (for the initial handshake) and  $\text{ServerAccept}_{k'}$  for each strict prefix  $k'$  of  $k$  (for the previous handshakes with pre-shared key). These events use as log a prefix of the log for  $\text{ServerAccept}_k$ . Therefore, if we have two executions of  $\text{ServerAccept}_k$  with the same log,  $\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys, i_0, \tilde{i}_k, i)$  and  $\text{ServerAccept}_k(x, \tilde{x}_k, \log_4, s\_keys', i'_0, \tilde{i}'_k, i')$ , then we have two executions of  $\text{ServerAccept}$  with the same log, which implies that they have the same replication index by (31), so  $i_0 = i'_0$ . For each strict prefix  $k'$  of  $k$ , we also have two executions of  $\text{ServerAccept}_{k'}$  with the same log, which implies that they have the same final replication index by (31) with additional index  $k'$  and additional arguments  $x, \tilde{x}_{k'}$ , so  $\tilde{i}_k$  and  $\tilde{i}'_k$  have the same  $(|k'| + 1)$ -th element, where  $|k'|$  is the length of  $k'$ . By (42), we have  $i = i'$ . So by combining these results, we have (43).

The same remark applies to the correspondences (33) and (35) as well.



**RESEARCH CENTRE  
PARIS**

2 rue Simone Iff  
CS 42112  
75589 Paris Cedex 12

Publisher  
Inria  
Domaine de Voluceau - Rocquencourt  
BP 105 - 78153 Le Chesnay Cedex  
[inria.fr](http://inria.fr)

ISSN 0249-6399